

NAVAL POSTGRADUATE SCHOOL

Monterey, California



19980210 129

THESIS

**DEVELOPMENT OF GRAPHICAL USER INTERFACE
STANDARDS AND PROTOTYPE FOR THE STUDENT
SERVICES DEPARTMENT OF THE MARINE CORPS
INSTITUTE**

by

Gerald L. Hehe

September, 1997

Thesis Advisor:

Magdi N. Kamel

Second Reader:

Dale Courtney

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September, 1997		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE Development of Graphical User Interface Standards and Prototype for the Student Services Department of the Marine Corps Institute				5. FUNDING NUMBERS
6. AUTHOR(S) Hehe, Gerald L.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Director of Marine Corps Institute, Washington Navy Yard, 912 Poor St. SE., Washington, D.C. 20391-5680				10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.				12b. DISTRIBUTION CODE
13. ABSTRACT (maximum 200 words) This research supports a year long Marine Corps Institute project initiated to migrate from a closed non-relational legacy system to an open client/server system architecture in response to many identified shortcomings of the current information system used by the Student Services Department. The objectives of this thesis are: (1) to identify a set of Graphical User Interface (GUI) guidelines for application development, (2) design and develop a proof-of-concept prototype that demonstrates the functionality of a relational database management system, and (3) exercise usability testing to validate the prototype functionality. Additionally, an object oriented visual development tool is used to develop the prototype application based on process and data modeling constructs. Implementation recommendations include: (1) adopting a continuous application development strategy based on modern concurrent process and data modeling constructs, (2) utilizing an object oriented visual development tool that compliments the target relational database management system, (3) utilizing the GUI guidelines identified during this research for future application development, and (4) applying usability testing to validate application functionality prior to implementation.				
14. SUBJECT TERMS Graphical User Interface, Department of Defense, Marine Corps Institute, Rapid Application Development, Relational Database Management System, Object Oriented Programming				15. NUMBER OF PAGES 133
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified		20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

Approved for public release; distribution is unlimited

**DEVELOPMENT OF GRAPHICAL USER INTERFACE STANDARDS AND
PROTOTYPE FOR THE STUDENT SERVICES DEPARTMENT OF THE MARINE
CORPS INSTITUTE**

Gerald L. Hehe
Lieutenant Commander, United States Navy
B.S., Northern Arizona University, 1981

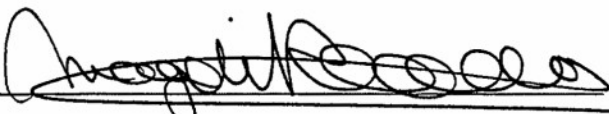
Submitted in partial fulfillment of the
requirements for the degree of

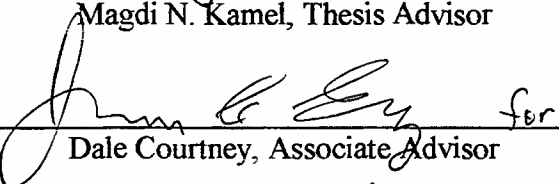
MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT


from the

**NAVAL POSTGRADUATE SCHOOL
September 1997**

Author: 
Gerald L. Hehe

Approved by: 
Magdi N. Kamel, Thesis Advisor

 for
Dale Courtney, Associate Advisor


Reuben Harris, Chairman
Department of Systems Management

ABSTRACT

This research supports a year long Marine Corps Institute project initiated to migrate from a closed non-relational legacy system to an open client/server system architecture in response to many identified shortcomings of the current information system used by the Student Services Department. The objectives of this thesis are: (1) to identify a set of Graphical User Interface (GUI) guidelines for application development, (2) design and develop a proof-of-concept prototype that demonstrates the functionality of a relational database management system, and (3) exercise usability testing to validate the prototype functionality. Additionally, an object oriented visual development tool is used to develop the prototype application based on process and data modeling constructs. Implementation recommendations include: (1) adopting a continuous application development strategy based on modern concurrent process and data modeling constructs, (2) utilizing an object oriented visual development tool that compliments the target relational database management system, (3) utilizing the GUI guidelines identified during this research for future application development, and (4) applying usability testing to validate application functionality prior to implementation.

TABLE OF CONTENTS

I. INTRODUCTION	1
A. BACKGROUND	1
B. SCOPE LIMITATIONS AND ASSUMPTIONS	3
C. OBJECTIVES	4
D. RESEARCH QUESTIONS	4
E. SCOPE AND METHODOLOGY	5
F. ORGANIZATION	6
II. BACKGROUND	7
A. MARINE CORPS INSTITUTE	7
B. CURRENT SYSTEM OVERVIEW	8
C. NEW SYSTEM ANALYSIS AND DESIGN EFFORT	10
1. Information Engineering Methodology (IEF)	10
2. Technology Architecture Planning	13
3. Data Modeling	15
4. Proof-of-Concept Prototype	15
a. Process Specifications	16
b. Data Flow Diagrams (DFD)	17
c. Database Schema	17
D. CHAPTER SUMMARY	17
III. GUI STANDARDS	19
A. EVOLUTION OF USER INTERFACES	19
B. ORGANIZATIONAL GUI STANDARDS	21
C. STRUCTURE GUIDELINES	23
1. Windows	23
2. Dialog Boxes	24
3. Menus	25
4. Toolbars	26
5. Pop-up Menus	27
D. INTERACTION	27
1. Command Buttons	28
a. Labels	28
b. Button Sizes	29
c. Grouping Command Buttons	30
d. Default Buttons	30
2. Option Buttons	31

3. Check Boxes.....	31
4. Text Boxes	32
5. List Boxes	33
E. PRESENTATION	34
1. Developing a Home Base.....	34
2. Flow of View.....	35
3. Grouping Data.....	36
4. Font Selection.....	37
5. Use of Color.....	38
F. CHAPTER SUMMARY	39
IV. IMMEDIATE ASSIST PROTOTYPE.....	41
A. IMMEDIATE ASSIST APPLICATION OVERVIEW	41
1. Designing an Application for the Student Services Department (SSD) Clerk.....	41
2. Proving the Concept	42
B. DEVELOPER/2000 ENVIRONMENT OVERVIEW	43
1. Background	43
2. Developer/2000 Basics	43
a. Introduction	44
b. Environment.....	44
c. Forms.....	46
d. Blocks.....	46
e. Triggers.....	47
f. Canvases.....	48
g. Structured Query Language (SQL).....	48
C. PROTOTYPES AND GUI DEVELOPMENT.....	50
1. Prototypes	50
2. GUI Environment	53
D. APPLICATION DESIGN AND IMPLEMENTATION.....	54
1. Process for Designing the Application.....	54
2. Opening Screen	55
3. Student Search.....	55
4. Enrolling a Student	56
5. Course Catalog/Enrollment.....	57
E. CHAPTER SUMMARY	59
V. USABILITY TESTING.....	61
A. COMPONENTS OF A USABILITY TEST	61
1. Types of Usability Tests.....	61
a. Exploratory Testing.....	61

b. Assessment Testing	62
c. Validation Testing	62
d. Comparison Testing	63
2. Test Settings.....	63
a. Simple Single Room	64
b. Modified Simple Single Room.....	65
c. Electronic Observation Room.....	66
d. Classic Testing Laboratory	67
3. Test Personnel and Roles	68
a. Test Monitor/Administrator.....	68
b. Data Logger.....	68
c. Equipment Operators.....	68
d. Technical Experts.....	69
B. DEVELOPING THE TEST PLAN.....	70
C. SELECTING TEST PARTICIPANTS	72
D. CONDUCTING THE TEST	72
E. SUMMARIZING THE TEST DATA	73
F. CHAPTER SUMMARY	74
VI. CONCLUSIONS.....	75
A. LESSONS LEARNED	75
1. Development Tools.....	75
a. Delphi	75
b. Developer/2000.....	76
2. Prototype Development	77
a. Process Modeling	78
b. Data Modeling	79
c. Rapid Application Development (RAD).....	79
3. Usability Testing.....	80
a. Developing the Test.....	80
b. Conducting the Test	81
B. FUTURE WORK	81
1. Security Features	81
2. Exam Management	82
3. Diploma and Certificate Issue	82
4. Transcript Generation	82
5. Program Enrollment.....	82
6. LOGAIS Communication.....	83
7. Help Desk.....	83
C. ACHIEVEMENT OF RESEARCH OBJECTIVES AND QUESTIONS.....	84
D. CONCLUSIONS.....	86

APPENDIX A. SCREEN SHOTS	89
APPENDIX B. MCI TEST PLAN.....	103
APPENDIX C. USABILITY TEST RESULTS FOR THE MARINE CORPS INSTITUTE.....	109
APPENDIX D. DEFINITIONS AND ABBREVIATIONS.....	115
LIST OF REFERENCES.....	117
INITIAL DISTRIBUTION LIST.....	119

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support of the Marine Corps Institute in funding the travel and equipment purchased during the research phase of this thesis.

I. INTRODUCTION

This chapter provides information on the purpose and content of this thesis. Section A discusses the background of the Marine Corps Institute (MCI) and the modernization project designed to update their Information System (IS). Section B describes the objectives of this research. Section C presents the research questions that will be explored. Section D discusses the scope and methodology of the thesis, and Section E describes the organization of the thesis.

A. BACKGROUND

The Marine Corps Institute (MCI) was established to "develop, publish, distribute, and administer distance training and education materials to enhance, support, or develop required skills and knowledge of Marines and to satisfy other training and education requirements as identified by the Commanding General, MCCDC" (MCI Mission, 1997). To accomplish its mission, MCI is organized into seven functional departments: education and operations, student services, information management systems, occupation specialty, professional military education, production, and logistics.

The mission of the student services department is to support the enrollment, grading and management of the Marine Corps distance education and training programs. In support of its mission, the student services department employs an automated information system (AIS) to automate the actions required to support a student in the MCI correspondence program, maintain student records, and produce necessary management reports. The automated system, known as the Marine Corps Institute

Automated Information System (MCIAIS), is a legacy system developed in the late 1970's. It runs on a Hewlett-Packard 3000 mini computer running the MPE/iX operating system. MCIAIS is written in HP proprietary language "Transact" and accesses a Turbo-IMAGE hierarchical database. As is typical of many legacy systems, MCIAIS suffers from many shortcomings:

- It has over 110 "spaghetti coded" programs that are difficult to maintain, modify, and upgrade.
- It does not have underlying data or process models.
- The programs have poor functionality, no statistical analysis capability, and limited "ad hoc" query capability.
- It utilizes a "closed" non-relational database.
- It does not support Graphical User Interfaces (GUI).

In response to these shortcomings, MCI initiated a modernization project to redesign MCIAIS using "open" system architecture (both hardware and software). In addition, MCI is also reviewing and redesigning the business processes to better support its mission and current advances in training and education. MCI contracted with the Naval Postgraduate School to perform an analysis and develop a business process reengineering evaluation and migration plan proposal. A team of students was selected by Dr. Magdi N. Kamel, Ph.D. to conduct the evaluation and prepare the proposal.

This thesis documents the development of a functional prototype based on the daily activities of the Student Services Department of MCI. The prototype was

constructed using established Graphical User Interface standards in a Rapid Application Development (RAD) environment. The prototype was then subjected to usability testing to validate the functionality of the prototype.

The research and development was conducted from August 1996 through August 1997. The complete project report is available as the following two technical reports.

- NPS-SM-97-001: Analysis, Design, and Prototype Implementation of a Contemporary Information System for the Marine Corps Institute, Preliminary Report (Kamel et al., 1997)
- NPS-SM-97-006: Analysis, Design, and Prototype Implementation of a Contemporary Information System for the Marine Corps Institute, Final Report (Kamel et al., 1997)

Other Naval Postgraduate School theses that cover related aspects of the modernization project include:

- Data model design: A Relational Database Model and Data Migration Plan for the Student Services Department at the Marine Corps Institute (Slaughter, 1997).
- Architecture model design: A System Architecture and Migration Plan for the Student Services Department of the Marine Corps Institute (Evers Jr., 1997).
- Process model design: A Business Process Model and Reengineering Plan for the Student Services Department of the Marine Corps Institute (Baden and Peters, 1997)

B. SCOPE, LIMITATIONS AND ASSUMPTIONS

The modernization team efforts focused on four primary areas.

- Conducting a thorough review of the business processes and developing a process model that graphically portrays the organization.
- Developing a conceptual data model and associated relational schema.

- Developing a technical architecture and migration plan in order to allow MCI to transition from a closed non-relational system to an open client/server-based Relational Database Management System (RDBMS).
- Developing a proof-of-concept prototype.

This thesis deals with the identification of GUI standards, prototype development and usability testing. Successful prototype development for the immediate assist clerk of the Student Services Department at MCI, requires the integration of the process model, data model and business rules developed by other team members.

C. OBJECTIVES

The objective of this research is three fold:

- Development of GUI standards for MCI,
- Development of a proof-of-concept prototype based on the functions of a selected segment of the Student Services Department, and
- Validation of the prototype via usability testing.

D. RESEARCH QUESTIONS

The primary questions to be answered by this research are:

- Can a clear set of GUI standards for the development of a series of integrated applications be identified?
- Can a proof-of-concept prototype based on the identified GUI standards be developed based on the process and data models created by the NPS project team?

Subsidiary questions answered are:

- Can an object oriented visual-based application development tool be used to generate a prototype using the GUI standards developed for MCI?

- Can usability testing determine whether the prototype meets the needs of the MCI telephone support clerks?
- Can the process and data models be successfully transformed into a working prototype demonstrating the functionality of the proposed RDBMS?

E. SCOPE AND METHODOLOGY

To fulfill the objectives of this research, a literature search is performed to identify established GUI standards. (Weinschenk and Yeo,1995) provides a format for using components in a consistent fashion. Application of the techniques described in this text results in the establishment of a set of standards for MCI.

A proof-of-concept prototype, designed to reflect a selected segment of MCI's Student Services Department (SSD), is developed using a RAD concept. This application is based on the process model designed for MCI and interacts with the data model constructed for this project.

Rubin reveals a clear and concise method for preparing for and the executing usability testing to validate the RDBMS and the RAD prototype. Usability testing of this project is conducted following the guidelines of (Rubin, 1994).

This thesis builds on the premises of the three previously mentioned efforts. The technical architecture is designed to support the implementation of the prototype and a series integrated application design efforts that follow.

Specifically, this thesis conveys the efforts of identifying GUI standards, developing an application embedding these standards into a prototype, and then

conducting usability testing to quantify whether the application demonstrates the functionality of the process and data models developed.

F. ORGANIZATION

The approach used to prepare this paper is predicated on identifying GUI standards, building a proof-of-concept prototype based on those standards, and verifying this prototype with usability testing. This thesis is organized as follows:

Chapter II presents an overview of MCI, its current legacy system, problems associated with the this system, and the new system analysis design efforts conducted by the NPS thesis students.

Chapter III introduces GUI standards. These standards are categorized into the functional areas of Structure, Presentation, and Interaction. The chapter describes the proper use of components and makes recommendations for their employment in application design.

Chapter IV provides an overview of the Immediate Assist prototype, the development tool used to generate it (Developer/2000), background on prototyping, and discusses the approach used by the NPS project team.

Chapter V introduces usability testing, its basic components, testing environments, and testing personnel along their responsibilities. The chapter also discusses the compiling of the completed data and preparing test results.

Chapter VI presents lessons learned, recommended future works associated with the prototype, and conclusions associated with the objectives of this research effort.

II. BACKGROUND

The purpose of this chapter is to familiarize the reader with the Marine Corps Institute (MCI) and its information system. The chapter is organized as follows: Section A discusses MCI and its charter. Section B overviews the current information system. Section C identifies the components of the new system analysis and design efforts, and Section D provides a summary of the chapter.

A. MARINE CORPS INSTITUTE

The Marine Corps institute was established to "develop, publish, distribute, and administer distance training and education materials to enhance, support, or develop required skills and knowledge of Marines and to satisfy other training and education requirements as identified by the Commanding General, MCCDC. To complete this mission, MCI is organized into six functional departments: education and operations, student operations, occupation specialty, professional military education, production, and logistics.

MCI is a large and dynamic organization. In order to provide education and training to Marines, MCI has focused on three main areas: developing educational materials, administering the materials, and maintaining logistical service related to getting the materials to the students. Recent changes in Marine Corps policies have increased the number of students enrolled in courses. The student operations department and the administrative requirements of its automated information system (AIS) is the focus of this chapter.

B. CURRENT SYSTEM OVERVIEW

The student operations department mission is to support the enrollment, grading, and management of the Marine Corps distance education and training programs. In support of its mission, the student operations department employs an AIS to automate the actions required to support a student in the MCI correspondence program, maintain student records, and produce necessary management reports. The automated system, known as the Marine Corps Institute Automated Information System (MCIAIS), is a legacy system developed in the late 1970's. It runs on a Hewlett-Packard 3000 minicomputer running the MPE/iX operating system. MCIAIS is written in Hewlett Packard (HP) proprietary language "Transact" and accesses a Turbo-IMAGE hierarchical database.

Operators currently interface with the database via a series of DOS character-based applications operating on top of Microsoft's Windows 3.11 operating system. These 110 "spaghetti-coded" applications interface with the Turbo-IMAGE database, providing users with data retrieval and display as well as limited data manipulation capability. As is typical of legacy applications, their use is not intuitive, requiring substantial training time for new users. In addition, these programs were not designed to integrate with other applications. No statistical capability exists to examine raw data or perform ad hoc queries. Applications are difficult to understand. As a result it takes several months of on-the-job training for an operator to become proficient enough to provide customer service without the aid of a supervisor.

No data model of the current database exists. Adding new entities or changing the database to support new attributes related to existing entities would be difficult, if not completely impossible, due to the current design. Processes that the database currently support have changed or been totally eliminated. The database administrator is constrained by the requirement to retain useless data, while data the organization wants to retain are being ignored due to the inability to store it.

Marine Corps directives for all Marines to enroll in MCI courses to enhance their personal and professional development has resulted in an overwhelming amount of new data for MCI to maintain. In addition, this policy also resulted in a new requirement for MCI to provide accurate and timely data to promotion boards. These boards review each candidate's record related to past and current performance related to MCI courses. Not only must MCI develop and maintain the courses, it must administer the exams and record the results to external agencies. As a result, the MCIAIS system is stressed almost to the point of failure.

Customer complaints regarding the inability to get materials ordered and delivered have steadily increased. The inability of MCIAIS to provide accurate data on stock quantities or to generate certificates and diplomas in a timely fashion is becoming a larger issue to the organization. Marine Mail, a program where Marines can directly contact the Commandant, has routinely included a large number of complaints directed at MCI and its inability to provide the support needed. Most of these complaints center on customer service issues and are a result of the shortcomings and limitations of MCIAIS.

Recognizing these emerging changes and the limitations of their current AIS, MCI contacted the Naval Postgraduate School (NPS) and requested an analysis of the current system and a recommendation related to modernization of the current system using contemporary architectures, methodologies, and tools.

C. NEW SYSTEM ANALYSIS AND DESIGN EFFORT

Five students from the ITM curriculum were selected to review MCI and the supporting AIS. The focus of the team was to define the processes which encompass MCI, analyze the existing architecture to determine if it could support MCI and its potential growth, develop a new relational database model in support of the required processes, and develop a Rapid Application Development (RAD) proof-of-concept prototype to demonstrate the success of the data and process models identified.

1. Information Engineering Methodology (IEF)

To accomplish its task, the project team adopted the Information Engineering Methodology to analyze the new system. Information Engineering (IE) is a methodology developed by James Martin (Martin, 1990). The methodology is best represented as a pyramid having seven basic stages: Information Strategy Planning (ISP), Business Area Analysis (BAA), Business System Design (BSD), Technical Design (TD), construction, transition, and production. The pyramid is also divided horizontally. Figure 2.1 depicts the IE pyramid. The left side represents data and the right side relates to an activity's tasks. The horizontal division within each of the stages is useful in dividing the required tasks among members of the development team. For example, the left, or data, side of the

pyramid is associated with the data modeler's tasks of identifying data subjects and entity types, modeling the relationships, and normalizing the entity records. Activity tasks, such as business area identification, process decomposition, and matrix development, fall on the right side. To identify the subsystems and their boundaries the process modeling team utilized this IE structured approach.

Information engineering is made practical by the use of a Computer Aided Systems Engineering (CASE) tool. It is important to choose a tool in which planning, analysis, design, modeling, and construction modules are integrated and share the same encyclopedia. The metadata in the encyclopedia resulting from the ISP study is a valuable asset and should be updated periodically. As the strategic goals or objectives of the business change, the information in the encyclopedia should also be changed. This allows the business model to remain current and available for periodic review.

The portions of the ISP and BAA stages of the information engineering methodology that pertain to activities include enterprise level analysis of MCI, business area analysis of SSD, and a To-Be information system model.

Enterprise-level analysis provides an overview of the organization. This overview is used by the top level managers and reengineering teams to decide how to proceed with a reengineering plan. The overview should not be too detailed. It is used to establish a broad overview in a short time. Detail will be added later during the business area analysis stage.

The top level might be thought of as being like an author planning a book and creating its table of contents. He surveys the overall contents of the book and divides it into parts and chapters. He decides which chapters

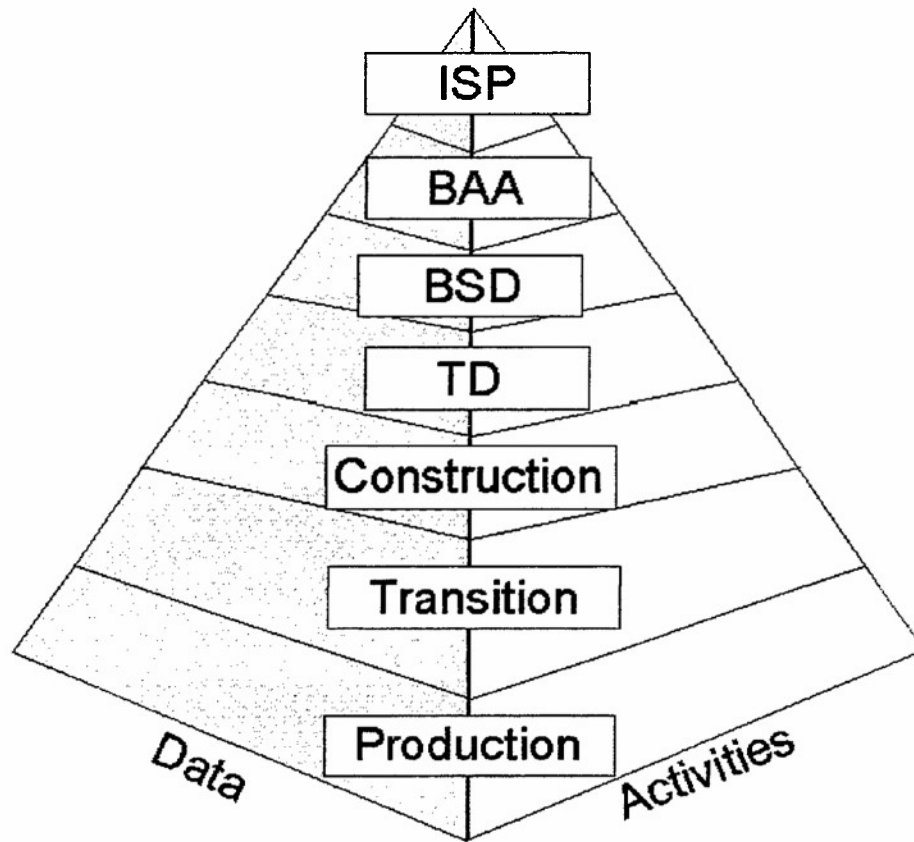


Figure 2.1 Information Engineering Pyramid about (Martin, 1990).

he should write first. Similarly, at the overview modeling stage he scopes out the overall structure and information needs of the enterprise, divides it into areas, and decides which area should first be analyzed in detail (Martin, 1990).

The overview information is stored in the CASE tool encyclopedia so that it can be updated over time and used for further analysis as detail is added. This CASE tool provides the functionality to include future growth or to alter the model in the event additional data becomes available at a later date.

To create an overview for the enterprise, data and process information must be integrated with the business strategy. Recall that the information engineering pyramid was divided horizontally with data information on the left and process information on the right. Thus an overview of the enterprise data is created when the data model information and the process model are mapped with the strategic information together along with its matrices. The matrices are then clustered to define the business areas. There are three steps to this process: (1) identification of organizational units, locations, functions, and entity types, (2) matrix analysis, and (3) identification of business areas.

As a result of clustering, seven business areas are identified for MCI: Personnel Administration, Ceremonial Support, Program and Course Management, Program and Course Development, Student Service Support, Warehouse and Distribution, and Information Systems management.

The resulting Process Specifications and Data Flow Diagrams (DFDs) will be the key components used by the prototype development team during the application design and implementation phase of the project.

2. Technology Architecture Planning

This area of integration represents specifying an architecture of future hardware and software platforms. Options ranging from maintaining the current hardware to upgrading existing hardware or completely replacing the current hardware must be considered. Spewak provides a four step guideline for reviewing and evaluating the technical architecture.

The first step is to identify technology platforms and principles. This step establishes the guidelines for the entire architecture development. The principles that will govern the development of the technology architecture must be based on trends and directions of the Information System (IS) industry. A literature survey is conducted to ensure critical industry options at least receive due consideration. After the principles are defined, the team compiles a list of the potential technology platforms for consideration.

The second step is to define the technology platforms and its distribution. With the principles of development defined, the team develops its strategy for the distribution of the applications and data. All business locations affected by the architecture is identified by location and function. The physical and conceptual location of the data is also determined in the distribution plan. Finally, a definition for the configuration of the technology platform is developed. This conceptual architecture must address the conceptual workstation (user access), the conceptual enterprise network (input/output, storage, and telecommunications), and the business systems architecture (implementing and maintaining applications and data of the enterprise).

The third step is to relate the technology platforms to the applications and business functions developed in the earlier components. In the planning guidelines above, the importance of linking the EAP to core business functions is discussed. In this extension of that philosophy, the defined technology platform must be related to the business functions that will use them and to the applications architecture that requires that technology.

The fourth step in technology architecture is to distribute the technology architecture. The documents defining the architecture are prepared in a clear, useful

format, then presented to executive management. The team discusses the potential gains and risks to the organization, data integrity and security issues, and implementation concerns. Briefings raise new issues for the team which must be considered for possible revision of the implementation plan. (Spewak, 1992)

3. Data Modeling

This effort consists of the construction of a data model which accurately captures the attributes of each entity identified by the process modeling team. An accurate data model depicting a developed database would allow the administrator the ability to adapt the database should future requirements dictate. With the utilization of a modeling tool, changes can be made and can instantly provide the required documentation needed to manage the new database. This allows for the synchronization of the database and model, with little or no disruption to day to day operations. Close interaction between the data and process modeling teams is critical.

4. Proof-of-Concept Prototype

After the process and data models have been constructed the application design and implementation effort can commence. Close integration of the application design team, the process modeling team, and the data modeler throughout the development cycle is critical. The development will combine the efforts of business process reengineering, technical architecture design, and the developed data model, combining them to produce a proof-of-concept prototype based on the functionality needed for the SSD customer support clerks for MCI.

The three most important components necessary for the RAD process are the process specifications, data flow diagrams, and the database schema. The process specifications provide the designers with a detailed document that describes the tasks to be created and embedded within the application. The DFDs provide a visual abstract, which the designer can use as a roadmap, to show where to get the data needed for the task and where to place the output. The database schema is the data modeler's tool for logically depicting the relationship between the entities described by the process modeling team.

a. Process Specifications

Process specifications describe the decision making logic and formulas that the application designer must use during the creative process of application implementation. The process specification should reduce the ambiguity of the task at hand, provide a precise description of what is to be accomplished, and validate the system design upon completion.

There are several types of process specification formats. The most commonly accepted are Structured English, Decision Tables, and Decision Trees. Capturing the task and transforming it into a sequence of events and precise inputs and outputs can be very involved. This effort is the most crucial part of the team's design effort. If the task is not completely defined or if a single component is overlooked, the result can be catastrophic.

b. Data Flow Diagrams (DFD)

Data flow diagrams illustrate data flows in and out of the system as well as the processing done on the flows. They also depict the distribution and storage of data. DFDs function as a roadmap for the application designer.

c. Database Schema

The database schema defines a database's structure, its tables, relationships, domains and business rules. A database schema is a design, the foundation on which the database and the applications are built. (Kroenke, 1995)

The database schema provides the application designer the information about the data available to the application. Specifics about how the data reflects the entity to which it belongs; how it is designed related to other entities; and the specific values described by the business rules of the organization are contained in this document.

To demonstrate the effectiveness of the process and data modeling teams, a RAD prototype is produced. This prototype is constructed using Graphical User Interface (GUI) technologies and incorporates a set of modified standards specifically designed for MCI. The new standards are incorporated into the future applications. The integrated programs will improve the users ability to learn new applications, make their introduction less problematic, and improve the proficiency of the users.

D. CHAPTER SUMMARY

This chapter discusses MCI's charter and some of the problems, political and external, that face the organization. To analyze the current system and identify potential

options aimed at migrating MCI to a new information system, a team of students from the Naval Postgraduate School has been established.

The MCI thesis team addresses the areas of technical architecture, business process reengineering, data modeling in depth. These areas are to be integrated and used as the basis for development of a selected features prototype to demonstrate the functionality of an RDBMS developed for MCI.

This thesis concentrates specifically on the area of GUI standards, GUI design, prototype generation, and usability testing associated with the proof-of-concept prototype developed and implemented for MCI.

III. GUI STANDARDS

Developers of Graphical User Interfaces (GUI) benefit from having a defined set of standards to use during the design effort. This chapter describes a set of guidelines for developing Graphical User Interfaces is organized as follows: Section A discusses the evolution of interfaces. Section B describes the need for developing organizational GUI standards. Section C discusses structure which is the framework used to create the interface. Section D describes how the user interacts with the application. Section E discusses the presentation of data, and Section F summarizes the chapter.

A. EVOLUTION OF USER INTERFACES

With the invention of modern computers, early applications and associated interfaces have been aimed at the computer programmer. Consequently the use of computers was limited to those who had an ability to develop complex series of instructions in a language specifically developed for a computer. Programmers would generate instructions with cables arranged on wire boards designed to move electronic values from register to register through complex adders and subtractors until the mathematical functions they had logically constructed provided the manipulation of the data into a form they wanted. Often the results were either simply displayed on a numeric panel or output in a plain-paper format.

With the advent of punch card technology and standardized compilers, the instructions devised by the programmer became portable. Programmers were able to develop standardized applications that are easily used but still understood only by

computer professionals with knowledge of high-level programming languages. Advances in these programming languages allowed standardized reports to be consistently produced in a format useful to the users.

Since there was a requirement to provide a direct interaction between a user and the computer, new real-time interfaces were developed. This ultimately led to the Cathode Ray Tube (CRT) display providing monochrome character display capability with a keyboard attached to allow operator interaction with the processor. Programmers familiar with the applications were satisfied with short prompts and blinking cursors to notify them that the computer was awaiting an input. Since this approach was not obvious to those not familiar with the specific application, a need for developing more consistent sets of responses and prompts became a requirement in interface design.

With the wide acceptance of the Personal Computer (PC), more users were utilizing the applications for a wide variety of computing requirements. Word processing, spreadsheet calculations, and some simple database processing were the primary applications available for the PC. The introduction of Apple Corporation's icon-driven desktop computer drastically changed man-computer interaction. Users now had a graphical interface with which to interact. With the release of Microsoft's Windows operating system, hundreds of new applications became available for users, ranging from computer games to graphical presentation applications to visual computer programming tools.

Today, the Windows-style interface is widely accepted among users, and its look-and-feel is the industry standard for application development. GUI, although a tremendous

improvement over character based interfaces, still requires a user-centered approach during its development. The standardization of data presentation, combined with a consistent method of interacting with the computer and consistent approach in structuring the application, has improved the design efforts and ultimately improved the quality of the applications available to the user today.

B. ORGANIZATIONAL GUI STANDARDS

A GUI environment has numerous advantages over a character-based interface. If designed properly the applications should be easier to learn, easier to use, and would improve operator performance and efficiency. Consistency is the cornerstone of the GUI design success. Components, screens, and menus should be used in the same manner and have the same functionality throughout the application. With an established set of standards, programmers will be able to develop a series of applications that not only look and feel the same, but make complementary applications more easily assimilated into the daily operations.

The GUI standards outlined in The Guidelines for Enterprise Wide GUI Design were utilized during the development of the prototype for MCI (Weinschenk, and Yeo,1995). These guidelines convey a great deal of expertise in designing user interfaces. It is highly recommended that MCI's project development team utilize these standards in conjunction with those listed in the following sections during the development of their future applications. The guidelines address three distinct areas of the design process: structure, interaction, and presentation.

Structure refers to the framework used to create the interface. It addresses guidelines related to the design of windows, dialog boxes, and menus. Additionally, it covers proper use of pop-up/drop-down windows and the relationship between menu-bars and command buttons.

Interaction refers to the user's communication with the computer. The user and computer communicate through the use of various controls. Choosing a control is not a haphazard process. Selecting the right control requires the designer to completely understand the task to be performed and how the user needs to interface with requisite data. Interaction guidelines address the proper use of list boxes, text boxes, spinners, etc.

Presentation refers to how data is displayed on screens and windows. This guideline instructs the developer on the preferred methodology to organize the layout of a screen. Additionally, this section demonstrates the proper way to create the flow of focus, vertically or horizontally, into screen design. Appropriate display of information can make a major difference in how useful the user perceives the interface to be.

Elements of structure, interaction, and presentation must be merged together into the application design effort to provide an interface that the user can understand. Successful screen design consists of developing a consistent pattern of using the components described in the previous paragraphs and overlaying them onto supporting object code that completes the defined tasks identified in the process model.

The purpose of developing GUI standards is to provide the in-house programmers with guidelines on how to consistently utilize visual components, develop logical screen flows, and improve the user's ability to quickly comprehend new application designs.

Interface design can have a major impact on the usability and usefulness of an application. A well thought out and executed design increases the user's understanding of its functionality and improve performance. The following sections recommend the guidelines outlined in (Weinschenk and Yeo,1995) and annotate specific recommendations that apply to MCI's application development effort.

C. STRUCTURE GUIDELINES

This area of GUI standards describes the way the application is presented to the user in the form of windows, dialog boxes, and menus. It allows users to interface with the application.

1. Windows

There are two options for window formats. Tiling or Cascading (Figure 3.1). Tiling windows are windows that are designed to be shown simultaneously. Cascading windows are designed to display on top of an existing window. Cascading windows are recommended because they allow the user to focus on the active window. Often, the task

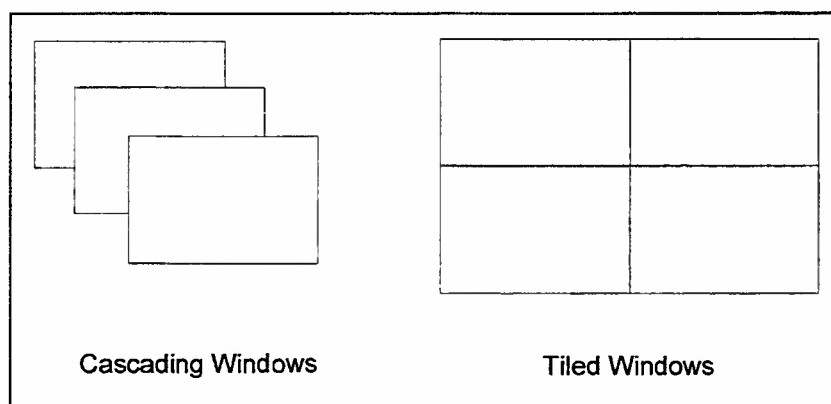


Figure 3.1 Cascading or Tiled windows.

requires a sequential series of actions to be completed. Under these conditions, the cascading windows work best. If a sequential flow is not required, and cascading windows would cover up essential areas of other screens, then the tiling method may be used.

Sizing of the windows is very important. The developer must avoid having the user scroll horizontally since it is not a natural viewing process. Using a larger window or breaking the information into several screens provides an alternative to horizontal scrolling. When supplementary windows are opened to expand on data presented in a previous window, the size of the window should be limited to the smallest size necessary to present or elicit the information of interest.

2. Dialog Boxes

Dialog boxes are one of the most important features of a GUI environment. By using a dialog box the developer has a greater chance of ensuring the user enters the data precisely, understands the sequence required to complete the action, or is warned of impending irrevocable actions.

There are two types of dialog boxes: modal and modeless. Modal dialog boxes stop the execution of the application until the user responds to a particular set of actions. Modeless Dialog boxes do not stop the execution of an application and may be deferred or even ignored since completion of the action is not required. Modeless dialogs may also be used to represent continuing work which does not impact the current process.

Dialog boxes should contain some visual cue as to the seriousness of the action that will occur in response to the prescribed message (e.g., red circle for danger or fatal

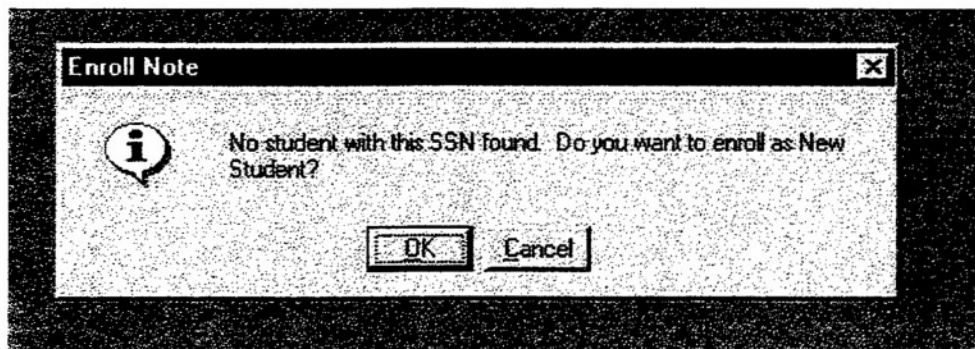


Figure 3.2 Modal dialog box actions, yellow for warning and blue for informational actions). Figure 3.2 displays a modal dialog box designed to communicate information to the user to help in decision making related to the enrollment of a student into a class.

Command buttons on a dialog box should be limited to OK, Cancel, Apply, and Help. Placement of the buttons should always be in the same order so users are not confused and inadvertently activate the wrong button. The default button selection should always be a non-destructive selection to minimize the possibility of the user making an irrevocable destructive action.

3. Menus

In order to move between functional areas in an application, developers may opt to use commands entered by the user or menu options selected from drop-down menus contained in menu-bars.

Menu-bars are usually grouped by specific areas like File, Edit, Help, View, and Tool options. For example, a File menu includes the capability to save, open, or close a

file. The associated actions are displayed with the first letter capitalized and a hot-key combination annotated (e.g., Ctrl+O). Care must be used in selecting hot-key combinations, and industry standards should be used wherever possible to facilitate user knowledge accumulated from widely used applications like Window 3.X or Windows 95. For example, Ctrl+X, Ctrl+C, and Ctrl+V are used commonly in Windows applications for cut, copy, and paste, respectively.

Single words are usually used to convey the meaning of actions in the menu-bar and drop-down menus to avoid confusing the user. Menu-bars must contain a single line of options so the selection of options is easy and straightforward. The menu-bar selection should activate the drop-down menu which contains the actions associated with the specific functionality it contains. Critical items should be placed at the top of the drop-down list, and similar functions should be grouped together with a separator line to provide the user a mental model of the grouped actions. For example, under the Edit segment the options Cut, Copy, and Paste are all related items used to edit text.

Menu options are a primary navigational tool for the user. Time and energy expended during the menu-bar and drop-down menus planning will be well worth the effort when end users need to move between the different modules of the application.

4. Toolbars

Toolbars allow the developer to collect frequently used options like drawing and formatting tools and display them as icons readily available with the click of a mouse. The developer must use care when creating toolbars to ensure that all the items displayed are

actually useful to the user. Toolbars should not be grayed out, and the collection of icons performing similar functions should be grouped together in a logical arrangement.

5. Pop-up Menus

Pop-up menus are menus that are displayed through special user actions (e.g., when the user clicks the right mouse button). Items presented in a pop-up menu should match the needs of the functions in which the user is actively involved. For example, if the user is involved with text manipulation, the displayed menu should activate items associated with that section such as a spell checking application or thesaurus.

Action Type	Proper Placement
Most frequent and critical	Command Button
Fairly frequent and across several screens	Toolbars
All actions: frequent, critical and infrequent	Menu-bar and drop-down menus

Table 3.1 Proper placement of actions from(Weinschenk and Yeo 1995).

Additionally, the items present on these menus must be commonly used items and not obscure or irrelevant functions. Table 3.1 outlines a methodology for determining the proper placement of actions.

D. INTERACTION

This category of standards relates to the way the operator interacts with the application. This section will discuss the proper use of command buttons and data boxes and their appropriate sizing and labeling.

1. Command Buttons

Command buttons are the means by which the user causes an action to be completed. These buttons are the primary method end users select to navigate from dialog box to dialog box, answer queries posed by the application, and return control back to the application after it retrieves necessary data from the user. Consideration should be given to their implementation.

When command buttons are programmed to launch major subfunction or program module, the developer should also include the ability to initiate this action from the menu-bar.

Developers should avoid overloading the screen with command buttons. No more than nine buttons should be placed on a full sized screen. When this number of buttons is exceeded the user can become overwhelmed and lose focus during the execution of the application. (Galitz, 1994)

a. Labels

Buttons labels must be clear and concise in describing the functionality of the action. A directive tone must be used to convey an authoritative attitude to the user to alleviate confusion. Book title capitalization rules should be used (capitalize the first letter of all major words). Unnecessary words should be omitted. Industry standards should be used where actions conform to those of existing applications. Table 3.2 provides a listing of several widely used buttons and their keyboard equivalents. Figure 3.3 provides an example of command buttons labeled in a clear directive manner.

Label	Action	Keyboard Equivalent
OK	Makes changes and closes the window	Enter key
Cancel	Does not make changes and closes the window	Escape key
Close	Closes the window when changes can't be made or canceled	C
Reset	Resets to the defaults, leaves window open	R
Apply	Makes changes, leaves window open	A
Help	Opens online help	H

Table 3.2 Standard labels for frequently used actions, from (Wienschek and Yeo 1995)

b. Button Sizes

Buttons should be sized relative to others in the grouping. If text labels vary to the point where different sized buttons are required, then every effort should be made to limit the number of unique sizes in order to keep the display uniform in

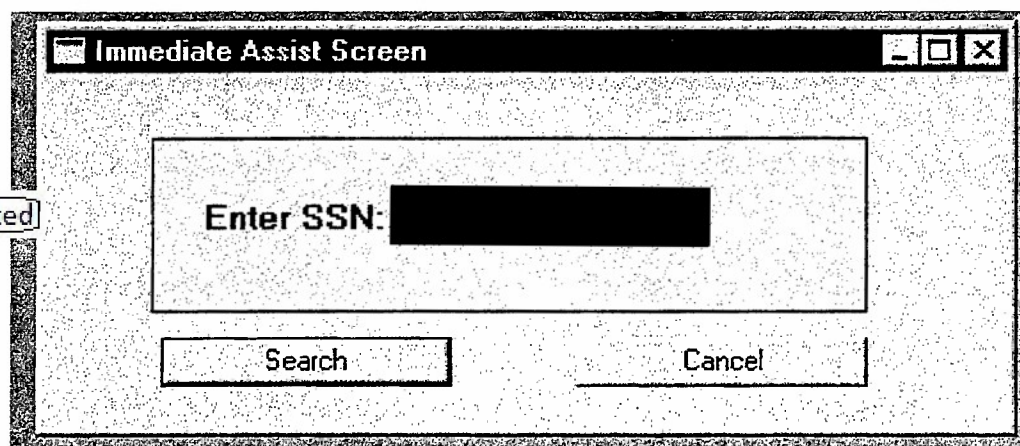


Figure 3.3 Label command buttons with clear directive meanings.

appearance. In the event that only one button is sized differently, the command should be reworded or the button placed where it does not visually distract the user.

c. Grouping Command Buttons

Command buttons that are functionally and logically related should be grouped together and bounded with a border. They should be equally sized and labeled in a uniform manner to avoid implying that one is more important or critical than the others. When designing screens, the grouping should provide ample white space in order to set off the buttons from the text or display boxes. Figure 3.4 depicts a grouping of command buttons.

d. Default Buttons

Developers should sparingly program default actions associated with buttons and always ensure that the action associated with the default is a non-destructive one. Users often hit the enter key and use arrow keys out of habit. Activation of a destructive nature must be programmed carefully and require the user to confirm the action prior to proceeding. Consistency in the use of default buttons is highly encouraged.

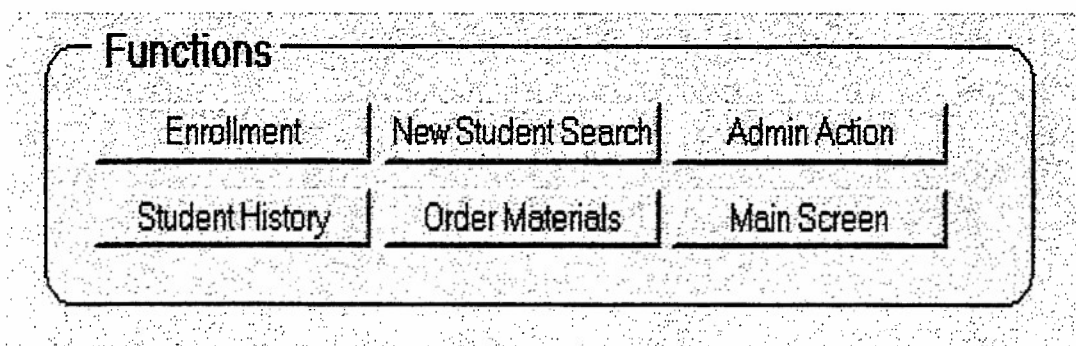


Figure 3.4 Command buttons grouped together and sized and labeled consistently.

2. Option Buttons

Option buttons are used where choices are mutually exclusive. Choices should be grouped together and labeled distinctly. The option buttons should be aligned vertically in order to make the options easily scanned by the reader. The developer should limit the selection to six items or fewer. Items should be ordered in some logical arrangement such as frequency of use or alphabetically. Binary conditions should be handled with a check box rather than a set of option boxes. Figure 3.5 shows the proper alignment and grouping of option boxes.

3. Check Boxes

Check boxes are used when the options are not mutually exclusive and the user may wish to select more than one option at a time. The same rules for option buttons apply for alignment, labeling, and ordering. Check boxes should be limited to 10 or fewer per group.

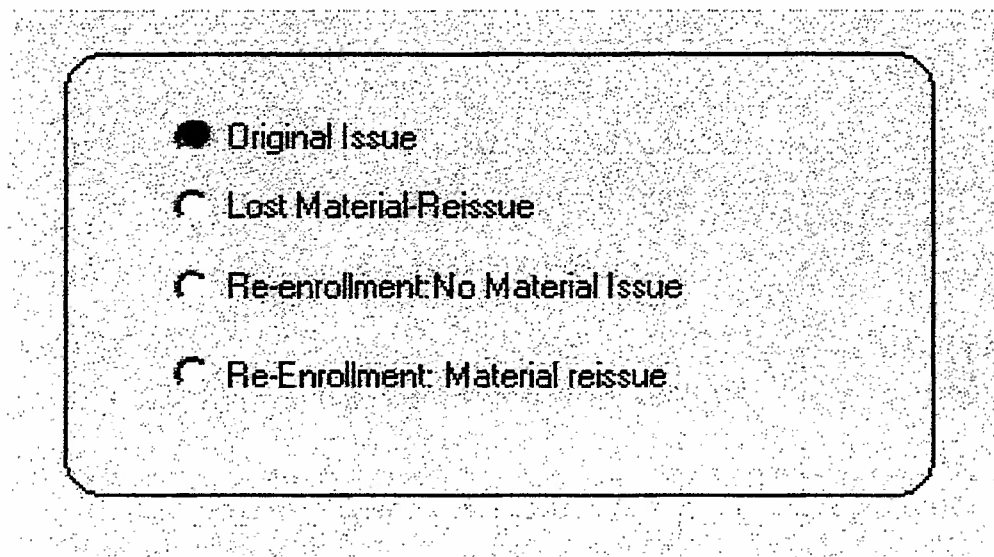


Figure 3.5 Proper alignment and grouping of option buttons

4. Text Boxes

Text boxes offer the user the ability to enter data into the application. If the field is protected and not editable, it should be temporarily grayed out. The developer should size boxes appropriately so that the user has an idea as to the size of the expected input. Where a large amount of data is to be entered and numerous text boxes are used, similar or relevant data should be grouped together (e.g., first and last name, or address fields). Fields should be left justified, and the number of unique margins on the screen created by the differing sizes of text boxes should be limited to two.

Labels should be placed to the left of text boxes and should end with a colon. Colons should not be used with frame names or column headings. All labels should be left justified. Avoid placing labels above text boxes. Figure 3.6 shows a data entry screen

Immediate Assist Screen

Update Student Data

SSN: [Redacted] PII Redacted

Last Name: Hehe

First Name: Gerald

Middle Initial: L

Rank: Lieutenant Commander

Grade: O4

Service Component: Navy

Figure 3.6 Proper alignment of text and list boxes.

showing the proper alignment of text boxes and labels.

5. List boxes

List boxes should be used when you want users to select predefined entries to preserve standardization and integrity in the database. List boxes also may be used as an alternative to option boxes when ten or more options exists. Where a very large pool of alternatives exists, consideration should be given to developing filters or subgroups to reduce confusion and make categorization easier. Often development tools allow for the use Lists Of Values (LOVs) in order to use codes for the storage of data but have the data displayed in verbose form. The developer must ensure the size of the list box is large enough to display all the possible selections, and if a dynamic list box (a type of list box that allows a new category to be entered in the event an existing category is not found) is used, that the size of the list box adjusts to accommodate the new entry. List boxes usually require slightly more space due to their expanding capabilities and should be placed in a position where they do not obscure other critical fields. Figure 3.7 demonstrates a typical list box displaying ranks of students.

Middle Initial:	L
Rank:	Lieutenant Commander
Grade:	Lieutenant Commander
Service Component:	Lieutenant General
	Lieutenant Junior Grade
	Major
	Major General
	Master Gunnery Sergeant
	Master Sergeant
	Master Chief Petty Officer
	Petty Officer First Class
	Petty Officer Second Class
Update Date:	te

Figure 3.7 List boxes should display all data contained in the field.

E. PRESENTATION

This section discusses guidelines for developing a homebase, designing a specific flow of screen presentation, grouping of data, font selection, and the use of color.

1. Developing a Homebase

Every application must have a screen which functions as the center point of the application flow. This screen is called a homebase. Since operators use this screen as the means to navigate to other areas of the application, its construction must be designed to clearly and logically convey a direction of the flow of the application to the user. The primary characteristic of a homebase is that users repeatedly return to this screen and use it to navigate to other areas of the application to complete the associated tasks inherent in the application. The developer should include command buttons or menu-driven commands supporting the user's return to this screen. Figure 3.8 provides an example of a homebase screen. Note that the user has a choice to select one of the five functional areas: Enrollment, Student History, Conduct a new Student Search, Order Materials, Admin Action, or to return to the Main Screen. These functional areas are the primary functions of the application. Activating any of those command buttons takes the user to a screen that accomplishes those tasks. Located on subsequent screens is a command button that returns the user to this homebase screen.

Immediate Assist Screen

Student Record

SSN: 526154472

Last Name: Hehe

First Name: Gerald

Rank: LCDR

USN

Navy

Address

Address: 22 Shubrick Rd.

City: Monterey

State: CA

Zip Code: 93940

Enrollment New Student Search Admin Action

Student History **Order Materials** Main Screen

Figure 3.8 Homebase screen with vertical flow.

2. Flow of View

Screen layout should be created to draw the user's focus vertically or horizontally across the screen. If a combination of both techniques is used on the same screen, the user may overlook data or may be uncertain as to which grouping, if several appear on the same screen, the command buttons correspond to. Vertical screens are most useful for data entry since most forms are constructed in that manner. Where data is required to be entered sequentially, the vertical design technique also works better. Horizontal screen designs work best when data can clearly be broken into subgroups, but the screen designer must take extra care in grouping the data and clearly include command buttons inside

defined borders with the appropriate text boxes. The developer should use one or the other technique and avoid combining them on the same screen. As a rule of thumb, command buttons are placed at the bottom of vertically oriented screens and on the side of horizontally oriented screens. Figure 3.8 shows an example of a vertically designed screen. Figure 3.9 displays an example of a horizontally designed screen.

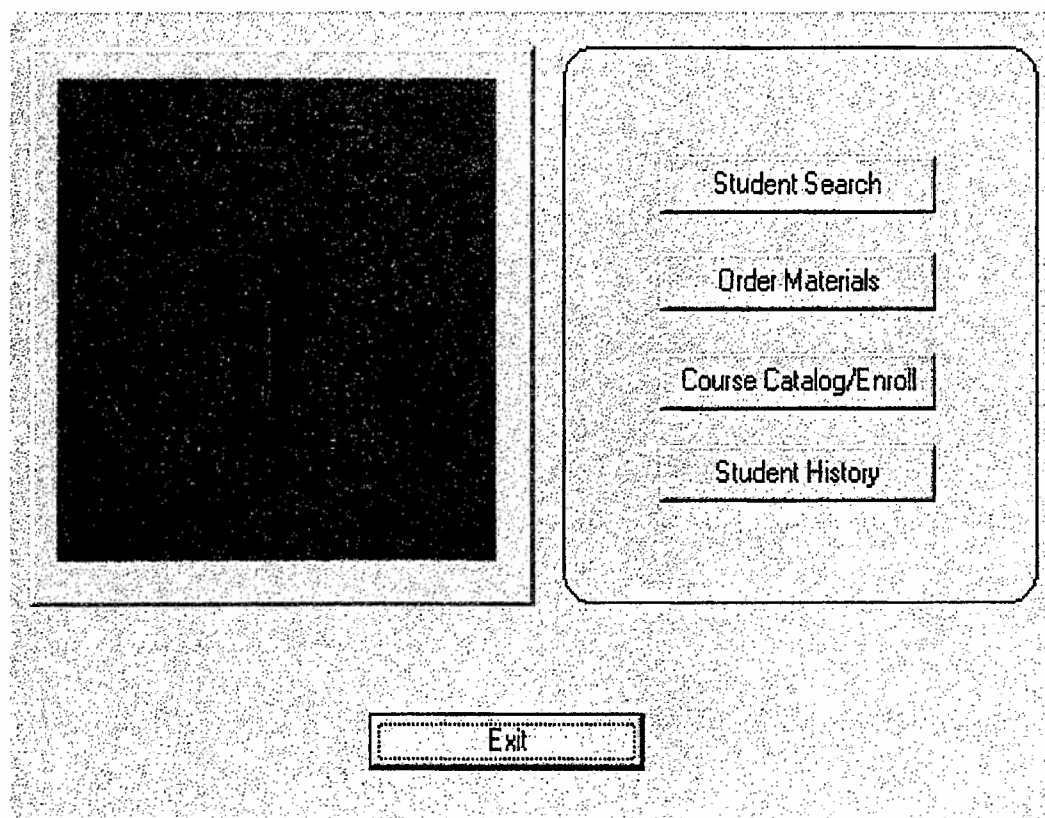


Figure 3.9 Horizontally designed screen

3. Grouping Data

To provide the user with a clear understanding of the data in question, the developer should use borders that visually group associated data together and use a descriptive group heading to set it off from other sections of the display. Use of white

space to show the groupings or to separate the command buttons associated with the data is highly recommended. Although borders are recommended, care should be taken to minimize the unique number of margins created by the groupings. The developer should attempt to align the borders wherever possible. Figure 3.10 displays a screen with address and phone number data grouped. Notice how the right margin of the phone grouping aligns with the right margin of the address grouping. Group headings should not include a colon. Figure 3.10 presents several grouped data fields for review.

SSN: [PII Redacted]

Last Name: Hehe

First Name: Gerald

Middle Initial: L

Phone

Commercial: 4086471343

DSN: 8782056

Address:

Address: 22 Shubrick Rd

City: Monterey

State: California

Zip Code: 93940

Figure 3.10 Grouping of associated data and use of descriptive headings

4. Font Selection

The use of a sans serif font is recommended for displays, since it is easier to read on the screen. Arial and MS sans serif fonts are highly recommended. Use of more than

three font weights in application design should be avoided. Although use of eight point font weight is acceptable, it is recommended that developers use ten point font weight as the minimum. This will allow development in smaller resolution environments like 800 x 600 or 1024 x 768 pixels per inch.

Use of italics or underlines in conjunction with the font should be minimized. Bolding should only be utilized for gaining the user's attention. Colored fonts are not recommended since a portion of the user community maybe color blind and unable to distinguish the change in color.

5. Use of Color

Color should be used judiciously. Initial design should start with a gray or light colored background. Color should only be used to attract the user's attention. When used too often, the ability to gain the user's attention with color will be diminished. When used, color should be consistent in subsequent screens.

Use of colors should be consistent with accepted norms since they can convey meaningful connotations (e.g. red should be used to denote danger or critical actions). Table 3.3 provides a listing of colors and their generally accepted meanings.

Color	Meaning
Red	Danger, stop, hot, or financial loss
Yellow	Warning or Caution
Green	Go or OK
Blue	Cool
Black	Financial profit

Table 3.3 Colors and their meanings in the United States, from (Weinschenk and Yeo 95)

Selecting a background color should be done carefully. Often developers attempt to provide a customized touch by selecting unique textures and color combinations as the application background. This may create more problems than it is worth if the texture obscures data or distracts the user. The developer should avoid the use of dark backgrounds and light text and be aware of blue and red combinations since they lead to eye fatigue and discomfort (Galitz, 1993).

F. CHAPTER SUMMARY

This chapter provides a short description of the evolution of the Graphical User Interface (GUI), a discussion on GUI standards, the three primary categories of GUI standards and their use in application design (structure, presentation, and interaction), and a discussion on components used to create the interface along with their suggested use.

IV. IMMEDIATE ASSIST PROTOTYPE

The objective of this chapter is to discuss the design and implementation of a proof-of-concept prototype of a selected Student Support Department (SSD) application. It is organized as follows: Section A provides an overview of the Immediate Assist application chosen for the proof-of-concept prototype. Section B discusses Oracle's Developer/2000 front-end development environment used to generate the prototype, Section C discusses prototyping and Graphical User Interfaces (GUI). Section D provides the user with an overview of the application design and implementation aspects of the prototype, and Section E summarizes the chapter.

A. IMMEDIATE ASSIST APPLICATION OVERVIEW

1. Designing an Application for the Student Services Department (SSD) Clerk

After defining the scope of the project and developing preliminary data and process models, the thesis team in agreement with the MCI organization decided to develop a proof-of-concept prototype in order to demonstrate the functionality of the proposed client/server application system.

Marines assigned to answer phone calls in a customer support roll at SSD complete a number of tasks supported by the Information System (IS). The most frequent are:

- Entering non-Marine students into the database

- Recalling historical data concerning course completion, diploma issue, and other course-related data
- Enrolling students into courses
- Collecting and entering data for course material distribution

These tasks require the clerks, known as immediate assist personnel, to interact with the constituent and database to manipulate data via a transaction code-oriented DOS-based application.

The clerks consult a number of reference books during their calls. These references are a course catalog, to identify courses and prerequisites, a logbook, to identifying transaction codes, and a logbook containing logistic memorandums related to stock levels on hand. The currency of these reference materials often becomes problematic when copies of the references utilized by the clerks are not updated.

Development of a GUI application, targeted at the daily operations conducted by the clerks, provides an opportunity for MCI to evaluate the merits of a new application by utilizing the established benchmark of the existing application as it relates to enrolling students, displaying course data, retrieving historical data, and ordering course material.

2. Proving the Concept

The purpose of developing the prototype is to demonstrate the capabilities of a client/server application system consisting of a graphical front-end user application on the client accessing a relational database on the server and based on the data process models developed by the project team. The purpose of the prototype is therefore two fold: 1) to validate the functionality of the combined process and data models, and 2) to show

improvements offered by a GUI environment over the DOS-based system currently employed.

B. DEVELOPER/2000 ENVIRONMENT OVERVIEW

1. Background

Application design and implementation efforts commenced with the evaluation of development tools. Team members reviewed available tools for ease of use, support of object oriented programming (OOP) and graphical interfaces, and the ability to interact with relational database management systems (RDBMS). Architecture concerns, such as the ability to store and modify applications on the server side and to minimize the configuration management associated with the large number of clients running the application, were also addressed in evaluating the tools.

Initial programming efforts were conducted utilizing the Delphi visual application development tool. In December 1996 MCI decided to conform with Marine Corps information standards and employ Oracle 7 as their supporting database. After the team considered the options available, the choice to develop the prototype application with Oracle's application tool Developer/2000 was made. This decision made the most sense when areas of security, interoperability, and integration of features were considered.

2. Developer/2000 Basics

This section will address the use of Developer/2000. An overview along with a discussion of the development environment and visual components is provided.

a. Introduction

Developer/2000 is an Interactive Application Generator (IAG) that allows the application developer to specify the look and actions of a program with a minimal amount of coding. This is accomplished by dragging and dropping components onto a repository and by linking objects with a few control commands to produce a fully functional application. This approach supports Rapid Application Develop (RAD) design concepts by producing an application much faster than using traditional methods.

Oracle's Developer/2000 provides a visual-based Object Oriented development environment which allows a programmer to create applications by associating objects, control, and relationships within a visual environment known as the Cooperative Development Environment (CDE). The following sections address the primary objects used in the CDE: Basetable blocks, control blocks, triggers, and how the developer ties them all together.

b. Environment

The basic interface for building an application is known as the object navigator. This environment is vehicle for the developer to create the interface with which the user interacts. The environment consists of:

- Forms
- Blocks
- Triggers
- Canvases

- Structured Query Language

Figure 4.1 shows the object navigator which the developer uses to develop the application. With this tool, the developer can add a new form, link existing forms together, create objects known as blocks and program the actions into triggers. This interface is the component most used by the application designer using Developer/2000.

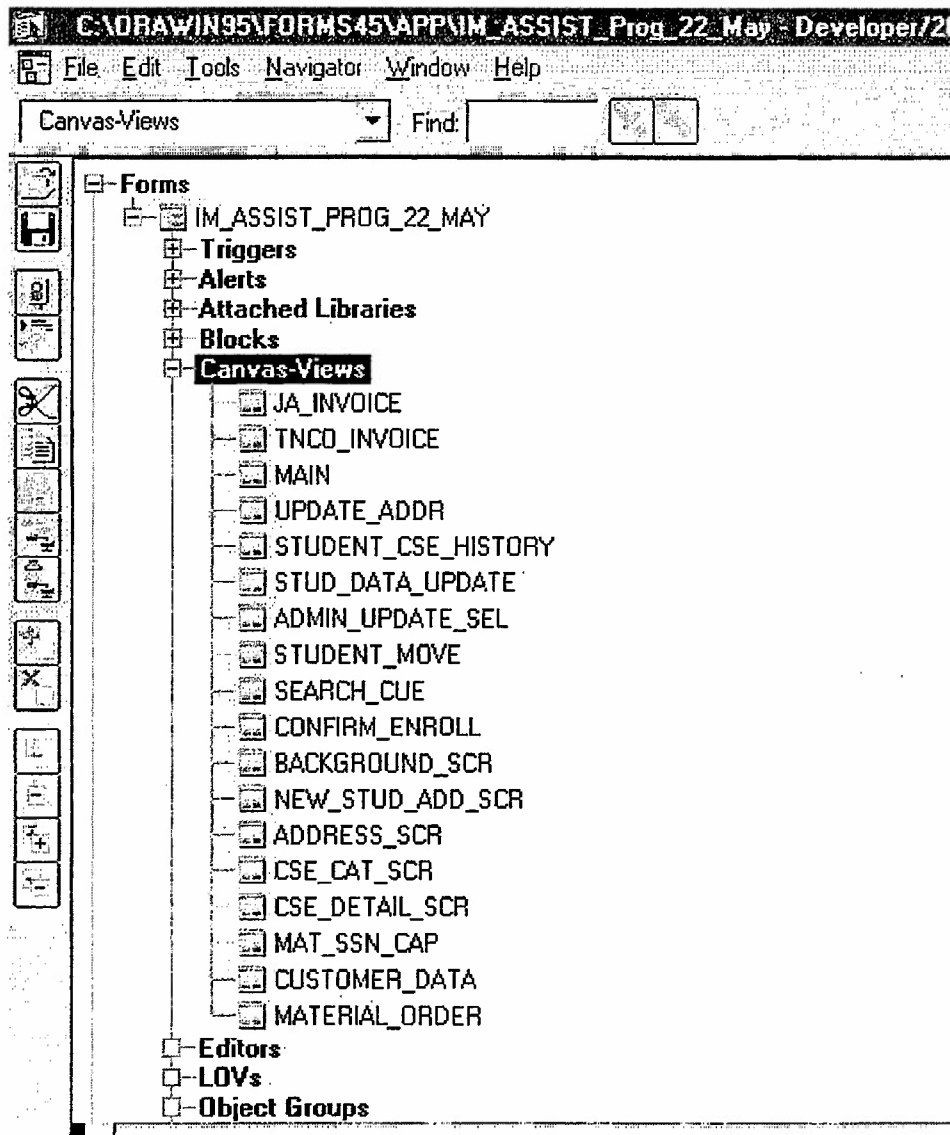


Figure 4.1 Developer/2000 Object Navigator.

c. Forms

The basic module is called a "Form". This form is a receptacle for storing all the items that are linked together to compose the application. Adding new items usually requires clicking on an Icon and filling in a dialog window.

d. Blocks

Blocks are the basic objects that the developer uses to interface with the database or to encapsulate instructions that manipulate the data. Blocks provide a mechanism for grouping related items into a functional unit for storing, displaying, and manipulating records. Just as tables in the database consist of related columns and rows, basetable blocks contain related items that display data records. There are two distinct types of blocks: basetable blocks and control blocks.

The most common block is the basetable block since it is the method used for connecting with a database. When creating the basetable block, the developer specifies on which table the block is based, which items are to be displayed, and which type of component is used to display it. Figure 4.2 depicts the dialog box used to create a block in the Developer/2000 CDE. The developer has the option to click on the select button in order to retrieve previously defined tables and canvases which will support the object or display the relationship.

Control blocks are a collection of triggers or may simply be a grouping of non-table related items. These Items may be used to structure control flow, reformat the data for a different display, or to enhance visual presentation of the data. Usually control

blocks channel the sequence of events and provide a capability to allow screens and components to provide functionality even if they are not specifically related to elements within the data base. For example, the prototype uses a control block to allow the user to input a SSN which is to be used as the basis for a query. Essentially, control blocks provide the domain for the programmer to manipulate data.

The combination of basetable blocks and control blocks embodies the basic concept of OOP. The developer builds the application to facilitate the entry, manipulation, calculation, or display of the data associated with the object blocks.

e. Triggers

Traditional computer programmers write an elaborate series of instructions that are executed from start to finish. In OOP the emphasis is on writing small segments

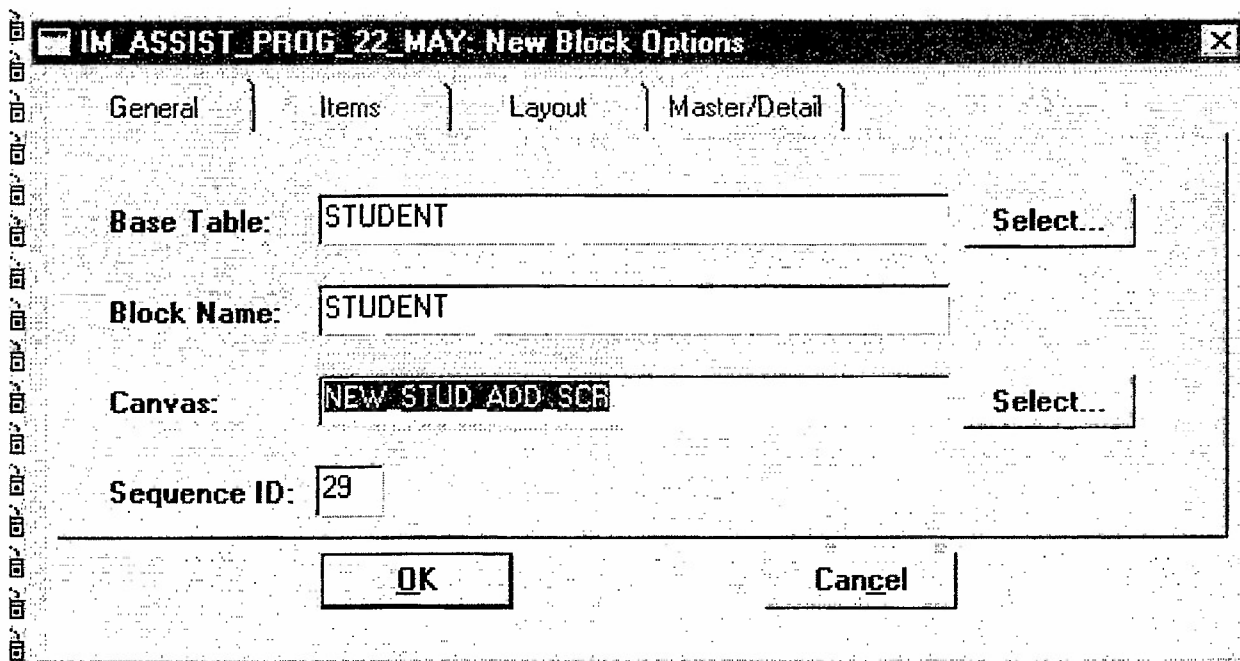


Figure 4.2 Developer/2000 Block development dialog box.

of instructions associated with a certain object and designed to be executed in response to a certain action. These small segments of instructions are known as “triggers” in the Oracle environment. These triggers are designed to perform a series of functions when they are activated or “fired.” Triggers may fire when a button is pressed, a list box is activated, or even if the control of the CPU is passed from one block to another. Programmers are limited only by their own creative abilities in defining triggers.

Creation of a trigger is associated with a form, a block, a component, or a prescribe combination of events. The trigger is developed from the object navigator in the PL/SQL editor. The developer clicks on the block, form, relationship, or canvas that the trigger is associated with and clicks on the trigger component to create the new item. A dialog box appears which includes an area to develop the supporting code along with the ability to test and debug the code. Figure 4.3 displays the PL/SQL editor.

f. Canvases

The visual component that exists specifically for displaying data is known as a canvas. It is the repository for all items that are displayed for the user. Canvases may be altered in size, color, and look. Their function is to display data in the items that are placed on them.

g. Structured Query Language (SQL)

The key component to any application is the ability to define the actions by writing instructions in some programming language for the computer to execute. The programming language used by Developer/2000 is Program Language/Structured Query

Language (PL/SQL). PL/SQL is basically standard SQL but includes enhanced features specifically designed for use in conjunction with the Oracle database. For client/server applications it is important to note that the connectivity between the client and server is done via SQLNet, and only standard SQL queries will be accepted. The enhancements of PL/SQL deal primarily with the manipulation of data returned by the SQL query. Figure 4.4 displays an example of PL/SQL code used to find a student record and display it.

```

PL/SQL Editor
Compile Save New... Delete Close Help
Type: Trigger Object: (Form Level)
Name: WHEN-NEW-FORM-INSTANCE

SET_WINDOW_PROPERTY(FORMS_MDI_WINDOW, WINDOW_SIZE, 560, 380);
Set_Window_Property(FORMS_MDI_WINDOW, POSITION, 0, 0);
Get_List_Values ('New_Stud_add.RANK', 'RANK.RankDesc',
                'RANK.Rank', 'RANK');
Get_List_Values ('New_Stud_add.Grade', 'Grade.Grade',
                'Grade.Grade', 'Grade');
Get_List_Values ('New_Stud_add.SERVCOMP', 'SVC_COMP.ServCompDesc',
                'SVC_COMP.ServComp', 'SVC_COMP');
Get_List_Values ('Non_mar_addr.State', 'STATE.STATEDesc',
                'STATE.STATE', 'STATE');
Get_List_Values ('Cust.State', 'STATE.STATEDesc',
                'STATE.STATE', 'STATE');
Get_List_Values ('CRSNO', 'CRS.CRSno',
                'CRS.CRSno', 'CRS');
Get_List_Values ('Student_Update.RANK', 'RANK.RankDesc',
                'RANK.Rank', 'RANK');
Get_List_Values ('Student_Update.Grade', 'Grade.Grade',
                'Grade.Grade', 'Grade');
Get_List_Values ('Student_Update.SERVCOMP', 'SVC_COMP.ServCompDesc',
                'SVC_COMP.ServComp', 'SVC_COMP');
Get_List_Values ('Update_Addr.State', 'STATE.STATEDesc',

Not Modified Successfully Compiled

```

Figure 4.3 PL/SQL editor.

C. PROTOTYPES AND GUI DEVELOPMENT

1. Prototypes

Prototypes can be classified in several ways. Each has a different focus and ability to display specific characteristics to interested observers. The main types of prototypes according to (Kendall and Kendall, 1995) are:

- Patched-up prototype
- Non-operational prototype

```
DECLARE
Counter      Number;
Alert_id     Alert;
button_pressed Number;
JPG_image_dir VARCHAR2(20) := 'c:/image/';
photo_filename VARCHAR2(20);
item_id ITEM;
Check_Stud_Exist EXCEPTION;
BEGIN
:Search_Cue.SSN_Pass:=(:Social||:Security||:Number);
:Student_Search.SSN_Entry:=:Search_Cue.SSN_Pass;
--Check to see if SSN represents active student record
  SELECT Count(*)
  INTO Counter
  FROM Student
  WHERE :SSN_Entry = StudSSN_ID;
--if the SSN is in the Student table retrieve data (Value should be 1)
  IF Counter <> 0 THEN
    SELECT StudSSN_ID,Rank,Grade,ServComp,StudLastName,StudFirstName
    INTO :SSN_Entry,:Rank_Displ,:Grade_Display,:Serv_Comp,:LastName,:FirstName
    FROM Student
    WHERE :SSN_Entry = StudSSN_ID;
      Select ServCompDesc
      INTO :ServCompDesc
      From SVC_Comp
      Where ServComp =
        (Select Servcomp
         From Student
         Where StudSSN_ID = :Student_Search.SSN_Entry);
  END IF;
END;
```

Figure 4.4 Program Language/Standard Query Language example.

- First of a Series prototype
- Selected Features prototype

The *Patched-up prototype* is designed to show a proof-of-concept. The actual processes and procedures utilized by the programmer may not be the most effective or efficient, but the ability of the application to demonstrate functionality is the primary purpose of the prototype design.

A *Non-operational prototype* is developed to show the system or product in a framework so the observer obtains a feel for the physical representation of the device and starts to envision the product in more than just a hypothetical situation. Often the purpose of the design is simply to demonstrate the anticipated size and shape of the device.

The *First of a series prototype*, as the name implies is a fully functional product which is designed for employment with revisions noted which are then quickly turned around and subsequent products delivered incorporating changes requested by the operator.

A *Selected features prototype* focuses on a subset of the requirements and demonstrates the product's ability to provide functionality to the user.

There are many advantages to utilizing a prototype to demonstrate the functionality of a project. Prototyping is most useful when the application definition is not well known, or when the end users requirements are not clear. Prototypes are well known for their ability to accommodate incremental changes in project scope or various changes in the organization's requirements. When requirements change, or the focus is redirected, a Rapid Application Development (RAD) process often allows managers to make better

quality decisions as to the most effective expenditure of resources. In the development of the prototype for MCI, there were numerous major changes including alterations in the processes being modeled, a change of the application development tool (from Delphi to Developer/2000) and a change in the supporting database RDBMS (from Interbase to Oracle 7). Even though these were major changes, the prototype concept easily adapted. Essentially, every element of energy expended was transformed somehow into the final prototype.

The prototype designed for MCI is a hybrid. It combines characteristics of the patched-up prototype and the selected features prototype. Enrolling a student into a course, retrieving data on a course, viewing student history, and providing a vehicle for ordering materials is demonstrated by the prototype. The prototype does prove that the developed data model is responsive and can be used to capture the data MCI clerks routinely need.

Management must fully understand the disadvantages of prototypes. Often the prototype receives criticism when management expects the application or product to be fully functional and ready for introduction into daily operations. There is great pressure to adopt the incomplete system into full operation. Only one specific type of prototype, the "First in a Series" prototype, is capable of being introduced in that environment. The first in a series prototype would best be used where an on-site programming effort is to be employed and the future releases are to be rapidly developed and introduced. Ultimately the MCI project personnel will be developing these types of applications and fielding them.

2. GUI Environment

A GUI environment has numerous advantages over a DOS-based interface. If designed properly the applications should be easier to learn, easier to use, and would improve operator performance and efficiency. Consistency is the cornerstone of the GUI design success. Components, screens, and menus should be used in the same manner and have the same functionality throughout the application. With an established set of standards, programmers will be able to develop a series of applications that not only look and feel the same but make complimentary applications more easily assimilated into the daily operations. Specifics regarding the GUI standards have been previously identified in Chapter II. These standards must now be implemented during the prototype design.

The proof-of-concept prototype will be demonstrated showing functionality related to the following SSD functions:

- Enrolling Students
- Building a view of course data (Catalog)
- Providing off-line order of material (link to future Log AIS system)
- Retrieving student history data

A comparison of MCI's current DOS-based screens and the GUI screens are presented in Appendix A. The figures in Appendix A present the following: the DOS application Main screen (Figure A.1), the GUI prototype Main Screen (Figure A.2), DOS application Enrollment screen (Figure A.3), GUI Prototype Enrollment/Course Catalog screen (Figure A.4), DOS application Student history search screen (Figure A.5), GUI

prototype Student History screen (Figure A.6), DOS student update screen (Figure A.7), GUI prototype Student Update screen (Figure A.8), the GUI prototype Homebase Screen (Figure A.9), and several additional screens discussed at a later portion of this report.

The prototype is evaluated by SSD clerks regarding the look and feel of the application along with any recommendations for functionality. These recommendations are collated and compiled into a report that is submitted along with a set of organizational GUI standards for the MCI program developer's use.

D. APPLICATION DESIGN AND IMPLEMENTATION

The Immediate Assist prototype was designed to allow the operator to enroll a student, display course data, allow the viewing of student history, and provide for the off-line ordering of materials. The design of the prototype was based on the data and process models detailed in (Slaughter, 1997) (Baden and Peters, 1997). This section will briefly discuss the design and implementation of the prototype.

1. Process for Designing the Application

An effective method for designing the application is to work closely with the process team that is decomposing the business areas outlined in the previous chapters. In the absence of working directly with the BPR team, utilization of the Data Flow Diagrams (DFDs) and process specifications is the next best option. During the development of MCI's Immediate Assist prototype, design began with a review of the DFDs and process specifications and was then followed by a chalkboard session. The chalkboard session allowed the project team to develop a visual sketch from the abstract requirements. From

there a mock up was constructed using Developer/2000 by placing components on the canvases in a way that supports the chalkboard discussion. The proposal was then presented to the group. Several iterations were conducted until the mock up gained the group's approval.

2. Opening Screen

The opening screen is a simple "switchboard" type concept. It provides a series of buttons that cause the prototype to launch into one of the four main processes: finding a student in the database, ordering material, displaying the course catalog, or enrolling in a course, and displaying student history, or exiting the program. To make the form more visually appealing, the MCI logo has been included on the screen. Figure A.2 displays the opening screen.

3. Student Search

In order to enroll a student or review a student's history, the prototype requires a valid SSN with which to query the database. When the user activates the Student Search button, they are presented with a screen designed to accept the SSN of a potential student. This screen is displayed in Figure A.13. An Oracle control block, called Search_Cue, has been constructed to search the database. When the user presses the search button, a trigger which encapsulates the necessary PL/SQL is fired and the database is queried for a matching SSN. The supporting PL/SQL related to the search button is shown in Figure 4.1. Finding a SSN related to a student requires more than running a query on a single table. The Marine Corps Total Force System (TFS) table also provides an input to the

process. If the SSN sought is not in the active student table of the database, the search must be expanded to search the MCTF_Personnel table. If neither table holds the SSN in question, the operator is notified via a dialog box (see Figure A.14) and then is allowed to enter the student data and cause it to be inserted into the database. If the operator needs to cancel the search, the cancel button trigger will fire, and the user is returned to the main screen.

4. Enrolling a Student

Figure 4.5 shows the data flow diagram which is the result of the process team's decomposition of the task called "enrollment". It illustrates the inputs to the process that must be provided by the operator, a validated student SSN and a validated course request. This assumes that the operator knows which data to enter. Additional data is needed complete the entry: the date of enrollment and the course completion date. MCI's business rule states that the enrollee has four years to complete the course. Discussion with the project team identified that the screen was static and that it would be more appropriate to combine the screen's functionality with the course catalog. The end result is that this screen is now presented as an enrollment confirmation. A button on the course catalog now allows the operator to select a course. The data needed is brought to the confirmation screen along with the system date and a trigger. These are used to calculate a date four years in the future for use as the course completion date.

5. Course Catalog/Enrollment

Figure 4.6 displays the data flow diagram for the “Display Catalog Information” process identified within the Provide Customer Servicing area. The same approach was used as listed in the previous paragraph: identify the inputs (Course/Program Information Request), and outputs (Course/Program Information).

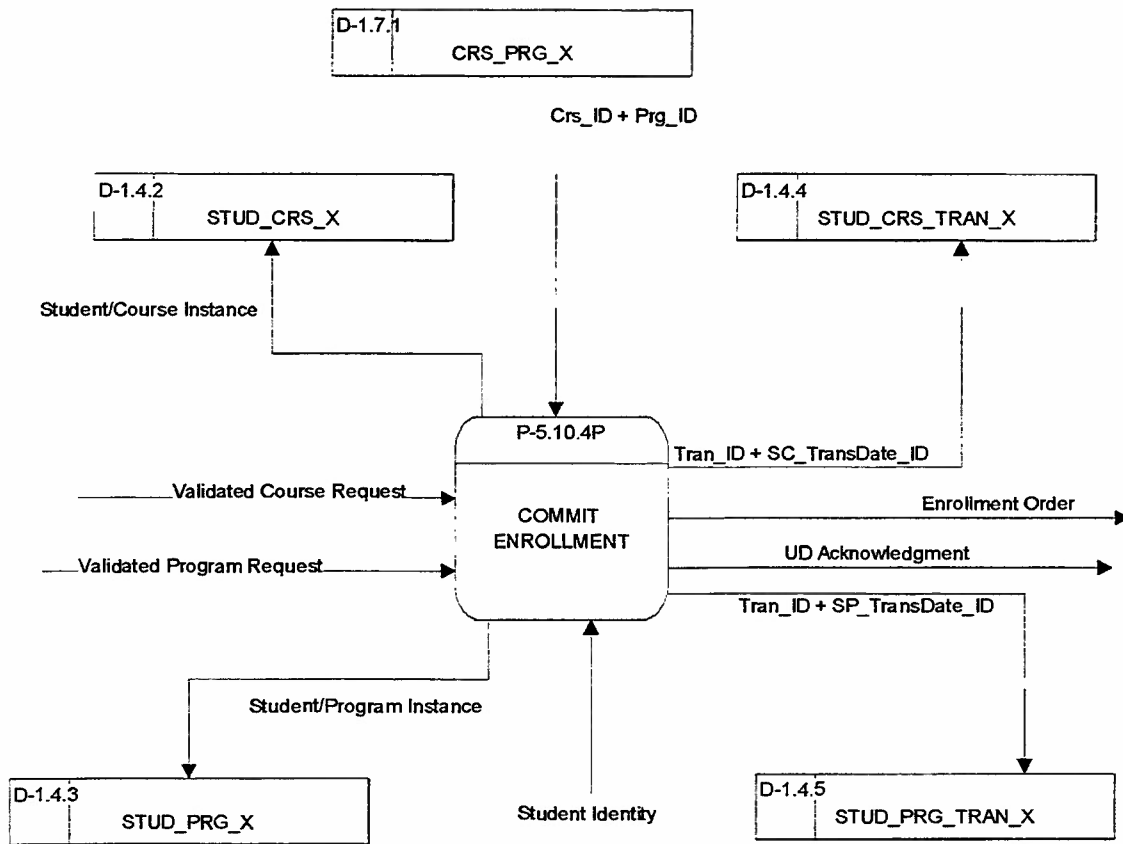


Figure 4.5 Enrollment Data Flow Diagram.

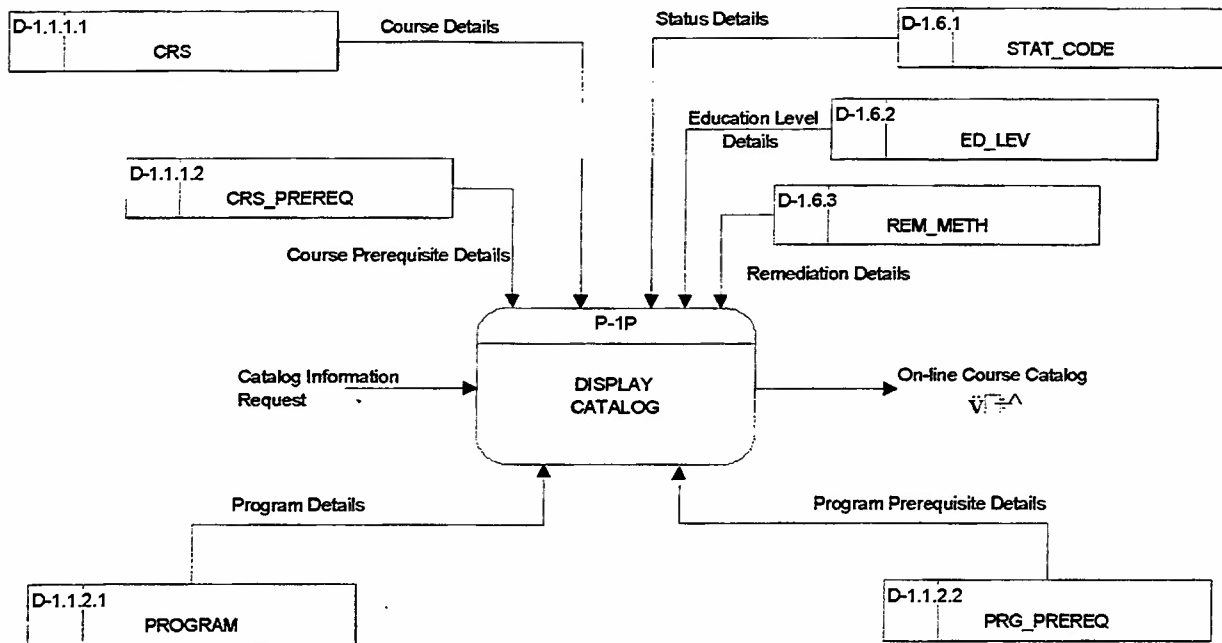


Figure 4.6 Catalog Dataflow Diagram.

The relevant tables were examined to determine the size, structure, and content related to Course Details, Status Details, Course Prerequisites, Program Details, Education Level Details, etc. Next, an attempt to conceptualize the screen design was conducted. Again a chalkboard session was held to collect ideas, and a mock up was produced within Developer/2000. Use of items and control components was guided by the standards previously noted in Chapter III. Careful consideration was given to not overloading the screen, proper selection of a single font, vertical flow, providing enough white space between the items, and not providing more components than is considered acceptable (no more than seven and no fewer than three is the accepted standard (Kendall and Kendall 1995)). Once the team was satisfied with the screen design, the actual coding effort commenced. The resulting Catalog/Enrollment screen is displayed in Figure A.4.

This process was repeated during the development of both the Student History and Material Ordering modules.

E. CHAPTER SUMMARY

This chapter provides an overview of the application design process related to the student services clerks and their daily operations. The primary objective of the design is to validate the process and data models used as the basis for the application and to develop a GUI-based application directed towards the daily operations of the student services clerks:

- Enrolling Students
- Building a view of course data (Catalog)
- Providing off-line order of materials (Link to future Log AIS system)
- Retrieving student history data

The Integrated Application Generator (IAG) tool selected for use in this project is Oracle's Developer/2000. A brief description of its environment and basic components called forms, blocks, triggers, and canvases is listed. These components are the building blocks that the developer uses to link together the data from the relational database management system and generate a functional application.

The implementation and design efforts predicated on the characteristics of the described prototypes are briefly outlined to provide some insight into the methodology used to generate the Immediate Assist Prototype for MCI.

V. USABILITY TESTING

During and following the development of the prototype, user-centered testing is conducted to ensure the usability of the design and user interfaces of the prototype. This chapter discusses the issues of user-centered testing and its. Section A describes the different components of a usability test. Section B outlines the development of a comprehensive test plan. Section C discusses the selection of test participants. Section D suggests a method for conducting the test, and Section E discusses the need for compiling data from the test.

A. COMPONENTS OF A USABILITY TEST

This section describes the components of a usability testing. It includes a discussion of the types of usability tests, test settings, and test personnel.

1. Types of Usability Tests

There are four basic types of software and application tests: exploratory, assessment, validation and comparative (Rubin, 1994). Each of these tests has a different purpose. The following paragraphs explain the different types of tests, and the advantages and disadvantages of each.

a. Exploratory Testing

The exploratory test is conducted early in the development cycle while the prototype is still in design stage. Using this method, the design team implements the functionality of the system as the process modeling team identifies the functionality of the

task. This provides user involvement in the development from the earliest point of design implementation effort.

The objective of the exploratory test is to evaluate the conceptual capabilities of a system or application design. This type of test requires close integration of the end users throughout the entire design and implementation process.

b. Assessment Testing

Assessment testing is the test most often used. It is usually conducted in the middle or later stages of the application development. At this point of testing the functionality of the process has been identified and embedded into the application.

The objective of the assessment test is to identify and expand the requirements and functionality of the application. This test is often called the “information gathering” test and is used in conjunction with an exploratory test methodology. The premise of this test is that:

- The user performs tasks rather than simply walking through and commenting on screens and pages.
- Test monitors lessen their interaction with the participants since there is less emphasis on the thought process and more on actual behaviors.
- Quantitative measures are collected and compiled (Rubin, 1994).

c. Validation Testing

Validation testing is normally conducted very late in the development cycle. This test centers on validating the usability of the application. This type of test is

important for the "first in a series" prototype since it helps identify problematic sections of an application as soon as it is fielded.

The objective of a validation test is to determine how well the product facilitates the completion of the defined processes and whether it meets the requirements of the user. Evaluation is based on benchmarks set by the organization prior to the development of the application.

d. Comparison Testing

The comparison test is not associated with any particular phase of development. It is used primarily to compare two or more systems. Comparison testing can be based on performance, ease of use, reliability, or a combination of any of these.

The purpose of this test is to evaluate one or more applications against another whose characteristics have been carefully benchmarked. This benchmarking allows design decisions to be made based on quantifiable data rather than emotional bias.

2. Test Settings

Rubin suggests four environments for conducting a usability test:: a simple single room setting, the modified single room setting, an electronic observation room, and the classic testing laboratory. Often the cost of establishing a laboratory for the express purpose of conducting the test precludes its use. A brief description, along with advantages and disadvantages of each environment, is provided in the following paragraphs. (Rubin, 1994)

a. Simple Single Room

This is the simplest and lowest cost environment in which to conduct a test. A small room configured with equipment needed to execute the application along with chairs for the participant and test monitor. Video equipment is placed unobtrusively to record the user actions during the test. Figure 5.1 depicts the simple single room environment.

(1) Advantages

- Least physical space required for testing of all environments.
- Monitors get a good sense of the testing since they are in close proximity.
- Fosters a sense of teamwork between monitor and participant.
- If required, the monitor can assist the participant during difficult tests.

(2) Disadvantages

- Monitor's presence may affect the participant's performance.
- Participant may depend on the monitor to help solve the problem.

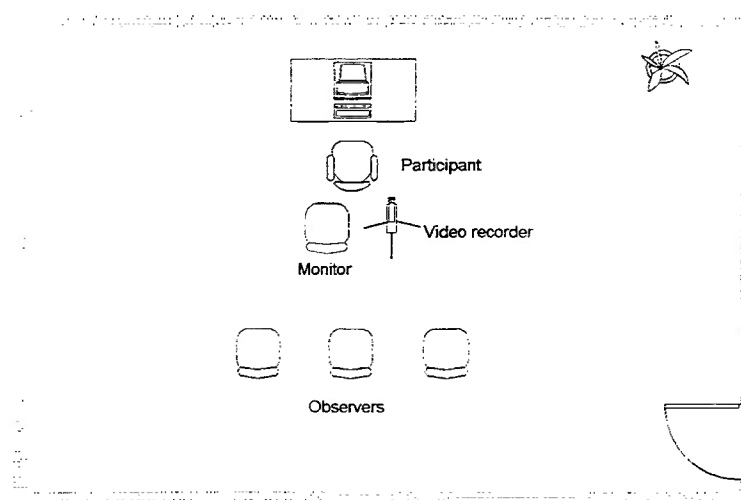


Figure 5.1 Simple single room setup. from (Rubin 1994).

- Minimal room available for additional observers to view the test.

b. Modified Simple Single Room

This setting takes the single room setup and moves the test monitor away from the participant. Instead of directly viewing the participant, the monitor observes the test via a TV monitor connected to a video camera that captures the participants actions.

Figure 5.2 displays an example of a modified single room setup.

(1) Advantages

- Participant is not influenced by the monitor's presence.
- Monitor is close enough to provide assistance.
- Participant is encouraged to think aloud.

(2) Disadvantages

- Monitor has a limited view of the proceedings.
- Monitor's presence can affect the performance of the participant.
- Limited space for observation of testing.

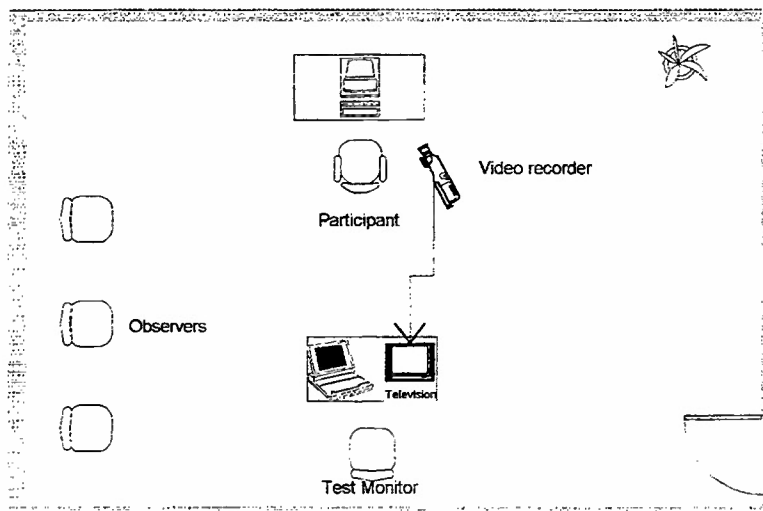


Figure 5.2 Modified Single room setup about, (Rubin 1994).

c. Electronic Observation Room

This room is used by organizations that develop a large number of applications and must test them prior to their release. It is the best setting for all the usability tests outlined in the previous section. This environment can be adjusted to accommodate tests with or without a test monitor present to assist the participant and provides for the most comprehensive data collection based on the positioning of the recording equipment. Figure 5.3 shows a typical Electronic Observation room setting.

(1) Advantages

- All the advantages of the single room and modified single room setup are in place.
- Ability to replay the test, via video playback, exists.

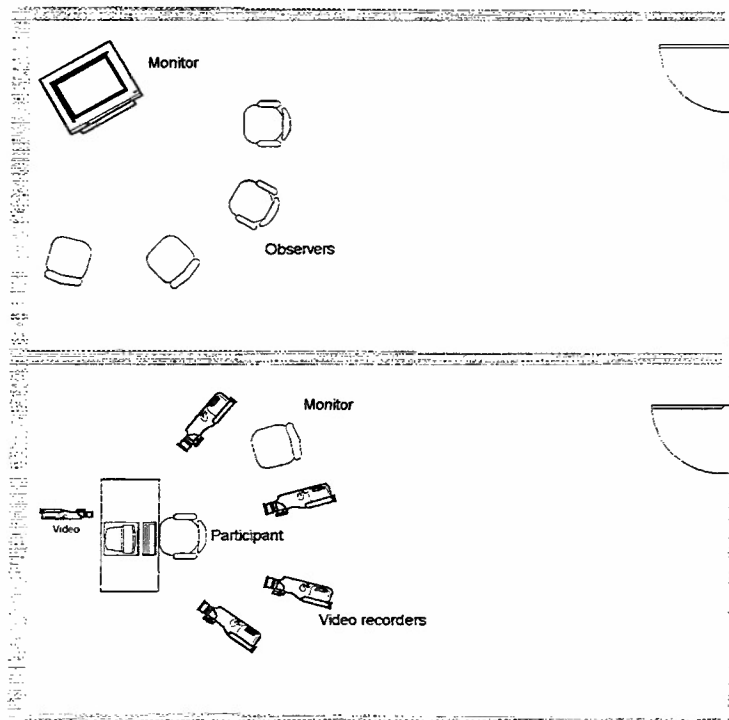


Figure 5.3 Electronic Observation Room setup, about (Rubin, 1994).

(2) Disadvantages

- Significant costs associated with construction and maintenance of the observation room.
- Quality of data is highly dependent on the placement of video equipment.

d. Classic Testing Laboratory

The classic testing laboratory provides an environment which allows the observers and test personnel to be completely separated from the test participant. This provides a better setting for validation and comparison tests but does not work well for assessment and exploratory tests. In the latter, close interaction with the participants is more important than analyzing their ability to successfully complete an assigned task.

Figure 5.4 depicts a possible Electronic Observation room setup.

(1) Advantages

- Best way to collect empirical data.
- Monitor does not affect test.
- Allows multiple monitors to view the screen, the participant, and any other aspect of the test which might be relevant to the design or implementation of the product.

(2) Disadvantages

- Room may be cold and impersonal.
- Camera placement is critical. Data may be obscured or lost due to improper placement.
- During exploratory testing, the control room setup may not offer any advantage since the participant may need the interaction of the monitor to complete some tasks.

3. Test Personnel and Roles

Personnel associated with a usability test have clearly defined roles and duties (Rubin, 1994). This section describes the functions of the test monitor/administrator, data logger, and equipment operator.

a. Test Monitor/Administrator

This is the most critical role. The monitor ensures the test material is in place and arranged, handles interaction with the participants, and ensures the results are documented and compiled. Interaction with the participant involves meeting and greeting, describing the test, and debriefing the test after it is concluded.

b. Data Logger

This role is designed to ensure that all pertinent data gets recorded properly. Data associated with each participant must be separated and compiled into a test report upon completion of the test. Good organizational skills are necessary for this role.

c. Equipment Operators

Personnel in the test room must make every effort to avoid distracting the participants during testing. Any adjustments to the video equipment should be done between tests or as quietly as possible by minimizing movement or sounds that would distract the test participants.

d. Technical Experts

Product designers, application development team members, and others associated with the construction of the prototype have a vested interest in the test. These technical experts are the most familiar with the prototype and its capabilities. To maximize the value of the feedback available from the participant, the technical experts observe the usability testing whenever possible.

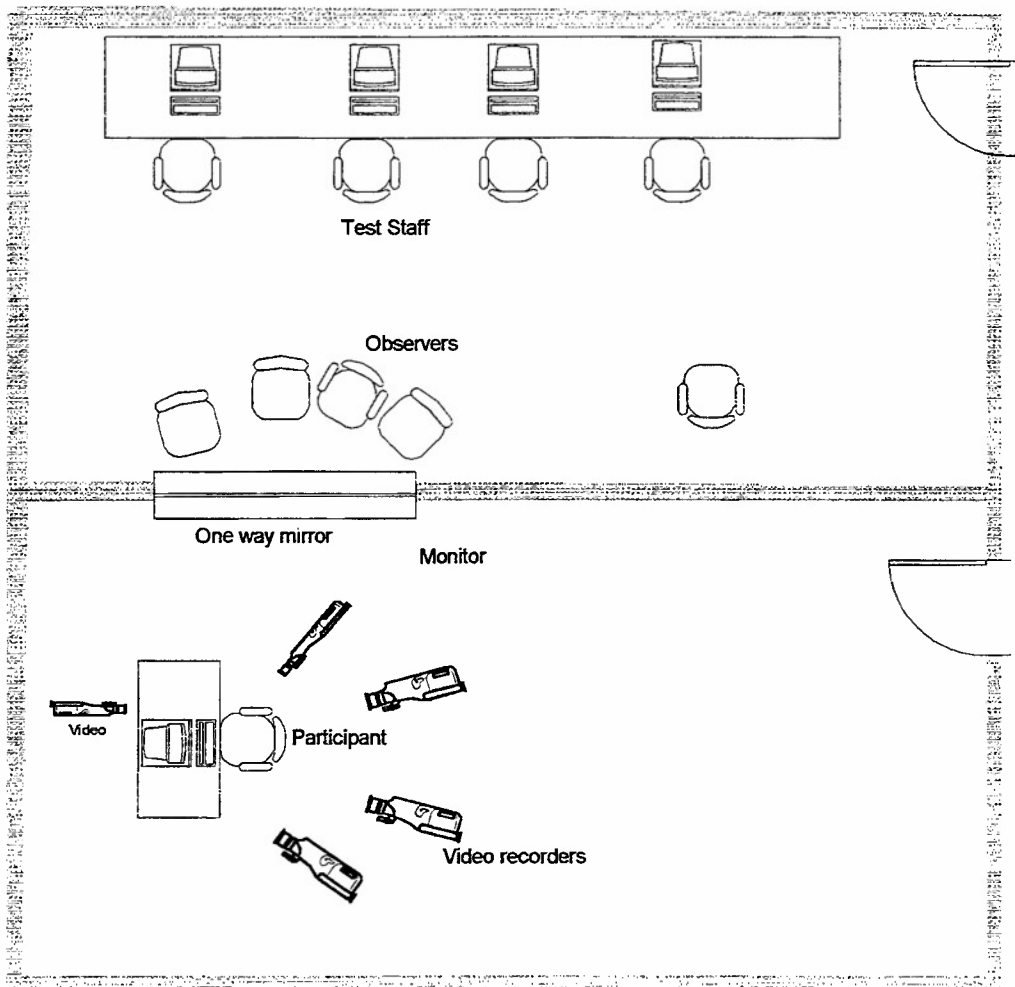


Figure 5.4 Classic Testing Laboratory setup, about (Rubin, 1994).

Technical experts can provide answers to questions that exceed the monitor's knowledge of the prototype. If they are going to be involved in the testing, they should understand the impact they can have on the participant and resist the temptation to answer questions during the testing procedure. Simple movements, sighs or gestures can impact the testing and the behavior of the participant. (Rubin, 1994)

B. DEVELOPING THE TEST PLAN

The goal of a usability test is to identify usability deficiencies that exist in computer-based systems prior to their release. The intent is to ensure the creation of products that are:

- Easy to use
- Satisfying to use
- Provide the functionality requested by the user (Rubin 1994)

To ensure the goals of a usability test are met, a test plan is developed, and a sample of users that reflects the actual population that will be using the product is selected. Test personnel must know what type of test is to be conducted, what type of feedback is needed, and what resources are available to them.

When the test plan is being constructed, test personnel must be told what information is being requested so that the proper type of test can be selected. If the focus is to be on validating initial concepts, the team will prepare for an assessment test and develop the supporting scenario and script. Testing of this type requires a less formal setting and is best conducted in a simple single room environment.

If the test is directed at verifying functionality of an application, the test personnel will most likely prepare for an assessment test. Since this test is more complex, relies on quantitative data, and depends on the environment available, more resources are required. The test monitor will have less interaction with the participant and is more concerned with the statistical data capture. This is the test most often utilized.

For situations where the evaluation of one application's performance is being compared to another, the testing team must establish benchmarks of the baseline application and then capture data on the performance to the other applications being evaluated. This means that testing efforts must be prepared for and conducted on all applications being evaluated as well as on the baseline system. Resources supporting this type of test will be the most significant.

After the test type is known and the environment selected, the testing personnel prepare the test script. This effort must take into consideration the anticipated test participants, their background, and experience levels. Test personnel must ensure the script clearly tests the objectives defined and supports the purpose of the test.

The test plan should contain the purpose of the test, provide a problem statement, and describe the methodology that the test will use during the event. Additionally, a structured set of tasks should be scripted, tested by the test staff, and included in the test plan. Appendix B provides a sample test plan that was prepared for the Immediate Assist prototype. This prototype was developed for MCI and tested on June 10, 1997.

C. SELECTING TEST PARTICIPANTS

Selecting the proper personnel to participate in the test is crucial. If the participants tested do not reflect the characteristics of the end user, the results of the test are useless. Participants should have the same educational background, age, and experience level as that of the end user. It is important that the characteristics of the test participants reflect the diversity of the user population.

The number of participants to test depends on several factors: degree of confidence in results, available resources, available participants, and time required to complete testing. For small organizations where the product is specifically targeted at a select group, testing the entire population of end users is highly recommended. This option was utilized during the testing of the Immediate Assist prototype at MCI.

D. CONDUCTING THE TEST

Conducting a usability test consists of preparing the test materials, setting up the testing area, briefing the participant on the test, performing the tasks as outlined in the test plan, and debriefing the participant.

After the room is prepared, the staff situated and the participant is briefed, the test can start. The monitor provides the participant with the task list or directs the participant to complete a set of tasks with a prepared script. Assistance from the monitor in completing the tasks varies based on the type of test being conducted (as discussed in the previous sections).

Deviations from the test script should be minimized. Data collection efforts and reports are predicated on the basis of having numerous participants completing the same set of tasks.

After the task portion of the test has been administered to the participants, a thorough debrief should take place. The participant has a great deal of information that can be captured by the monitor and data loggers. Questions related to what the participant liked and disliked about the product are valuable. More importantly, questions related to what functionality could be added or modified must be explored.

After a verbal debrief, participants should be given a questionnaire or survey to quantify their feelings about the design, performance, or implementation of the functionality. Participants should have access to the prototype while filling out the questionnaire. This relieves them from having to recall past actions, provides a more accurate assessment, avoids having the participants recalling the product from memory, and provides for a better critique of the system.

E. SUMMARIZING THE TEST DATA

After all the participants have completed the test, filled out the questionnaire and completed the debrief, data must be compiled into report to determine the success of the application design. Appendix C contains the test results obtained from the usability testing conducted on the Immediate Assist prototype designed for MCI. The report is designed to concentrate on two distinct areas:

- Identify specific weakness of the design implementation.

- Provide directions to management for future design efforts.

Weaknesses in design efforts that are identified during the test will assist the designer in future. This is very useful in the iterative development process.

Management is interested in the overview data. They can use the results in the life cycle management process to assist in allocating resources or making decisions as to the future of the project as a whole.

Objectives identified in the test plan should be addressed by the questionnaire. Survey and statistical data, if collected, should be compiled and presented to support decision making as to whether or not the application design effort succeeded in its attempt to meet the user's requirements.

F. CHAPTER SUMMARY

This chapter described components of usability testing. In order to properly evaluate a potential design implementation, the organization must decide on what type of test to conduct, select an testing environment that can be supported, and systematically conduct the test. The results of a usability test can improve the project team's ability to develop applications that meet user demands as well as provide management with valuable information related to future development efforts.

VI. CONCLUSIONS

This chapter details some lessons learned during prototype development as well as recommendations and conclusions. Section A of this chapter discusses the lessons learned during the project life cycle. Section B provides recommendations related to the future development on the Immediate Assistance application prototype. Section C addresses and answer the research questions and objectives presented in Chapter I. Finally, Section D provides some conclusions.

A. LESSONS LEARNED

The project team gained a wealth of insight into application design and implementation processes by defining Graphical User Interface (GUI) standards, developing the prototype, and conducting a usability test. Construction of a prototype in a RAD environment provided valuable experience in evaluating development tools evaluated, prototype development, and usability testing.

1. Development Tools

This project provided the NPS team the opportunity to examine two visual-based application development tools. Boreland's Delphi 2.0 and Oracle's Developer/2000 were used during the project development.

a. Delphi

Delphi was the first tool utilized by the team during the project. It provided an intuitive interface and the supporting language, Object Pascal, was relatively

easy to learn. Project team members quickly became proficient with the tool. However, several weaknesses with the Delphi application tool were identified. Although Delphi supported client/server interaction, the object code for the application required installation on each client. This was deemed to be a major deficiency. Changes to the application could not be made on the server side, and instead needed to be installed on each of the clients. This cumbersome procedure makes updates and configuration changes difficult. Additionally, the application was large and consumed significant storage on the clients, negating one of the benefits of client/server architecture.

b. Developer/2000

In December 1996 MCI decided to conform with newly released Marine Corps directive standardizing software usage throughout the Marine Corps. This changed the target database from Interbase to Oracle 7. After reviewing the incorporated security and integrated features of Oracle 7, the project team decided to further examine the Oracle visual Integrated Application Development (IAD) tool known as Developer/2000.

Designed by Oracle, Developer/2000 is marketed to support front-end application development for Oracle 7 databases. Developer/2000 provides for the storage of the application forms and modules on the server-side of the client/server architecture. The clients only need a small integration application called "Run Form" to be placed on each client which calls the necessary modules from the server and then executes the application. Program maintenance, upgrades, and modifications can be made one time on the server-side and all clients will instantly have access to the modified application. This

along with the built in security features that Oracle incorporated into Developer/2000 made it the best choice for the prototype development.

Learning to develop an application in the Developer 2000 suite was more difficult than with Delphi. Knowledge and proficiency was obtained by hands-on experience during prototype development. The learning curve associated with this complex tool was initially flat, but after basic skills were mastered the team proficiency increased quickly.

Developer/2000 training courses were available at NPS via video-teleconferencing in a distance learning classroom. These courses were not very beneficial because the material was presented on a teleprompter, was not clearly visible, and barely demonstrated basic skills. No hands-on training was available using this medium. Advanced Instructor Led Training (ILT) is available and recommended for any future projects interested in using Developer/2000. Time constraints precluded the NPS team from attending this type of training scenario.

2. Prototype Development

The enterprise-wide analysis identified the functional subsystems of MCI. The subsequent business area analysis focused on the student servicing functions. An integrated approach provided the team an opportunity to begin prototype development in a RAD environment. Each of these functions yielded significant lessons learned which were important for the actual prototype development.

a. Process Modeling

The functional areas of MCI were decomposed and analyzed to identify business areas. The group evaluated MCI processes across the organization to formally construct an enterprise process model. This effort was important to ensure activities currently conducted by MCI are understood by the modeling team.

The process modeling team identified the boundaries of the subsystems. Decomposition of the Student Services business area lead to the construction of the "As-Is" process model. The As-Is process model was then reengineered. Each process was evaluated to ensure it added value to the overall function of providing service to the customer. This effort produced the "To-Be" process model. An information system model was produced where the manual processes of the To-Be model were eliminated. A set of Data Flow Diagrams (DFDs), the corresponding process specifications, and a clustered matrix for the use by the prototype development team resulted.

The Clustered, or CR matrix identified candidate applications which the prototype development team would need to develop. This matrix also identified the interaction required between internal and external entities.

The process modeling identified the requirements that the application would to meet. A clear understanding of exactly what was to be accomplished by the application is now available to the prototype development team.

b. Data Modeling

A logical compliment to the process modeling effort was a clear definition of the entities with which the subsystem must interact. Data modeling efforts were directed at identifying and defining the attributes of the data subjects needed for the successful execution of the processes. The express purpose of the data modeling effort, from the point of view of the application development team, was to develop the framework that would act as a repository for the information needed to fully define the entities and to make it available in a RDBMS.

c. Rapid Application Development (RAD)

The purpose of RAD is to enable a prototype development team to quickly take the users requirements, as defined by the process and data models, and build a prototype that successfully integrates the data repository developed in accordance with the newly developed relational data model.

Visual-based Object Oriented Programming (OOP) tools allow quick generation of complex prototypes. End users can see the results and recommend improvements which can be rapidly incorporated. This iterative process makes RAD a valuable technique. It reduces the normal life cycle of application development. Effective integration delivers a better product to the user in significantly less time.

Prototype implementation for MCI demonstrated that a RDBMS could better support their needs than the existing flat file system. Additionally, the prototype

gave them the confidence that they could successfully migrate from a legacy hierarchical DBMS.

3. Usability Testing

Contemporary software development methodology is focusing on user-centered design. Testing to ensure the application actually meets the user's requirements and provides an interface that the user can understand is becoming a primary concern. To accomplish this goal, increased effort must be directed at the development of the usability test, conduct of the test, and the evaluation of the from the user's perspective.

a. Developing the Test

A usability test should not be a scientific experiment designed to prove a hypothesis. It should be directed at evaluating whether the objectives of the product can be met quickly, accurately, and with minimal investment in external resources (e.g., training courses or manuals).

Development of a test plan is the most important segment of usability testing. The test must have a clearly defined objective. The test itself must not be considered a success or failure by the results of prototype evaluation. The test must be evaluated on its ability to provide a controlled atmosphere, generate constructive feedback to the application developer, and by the participant's ability to become involved in the evolution of the application.

b. Conducting the Test

Successful testing relies on selecting the best environment, preparing the test personnel, selecting participants (that reflect the target population), and eliciting feedback from those participants. Once these components are organized, the participants evaluate the application.

Test personnel should administer the test without interjecting bias into environment. When conducted properly, the participants are oblivious to the test personnel presence during their evaluation of the application.

B. FUTURE WORK

The process and data models as well as the prototype have been delivered to the MCI project manager. Recommendations for the future application design efforts follow.

1. Security Features

The Immediate Assist prototype did not include security features, although the RDBMS provides for control of access. The first enhancement should be an access-control feature designed to allow only authorized personnel access to the RDBMS. Each role, (e.g., clerk, registrar and database administrator) should be assigned privileges and access to the tables commensurate with the functions they perform. For example, clerks should have read/write access to the "student" table to enroll new students but only read access to the "course" table. Each role must be carefully balanced by a database administrator to ensure the granted access privileges/permissions are correct. Additionally, each user should have a unique password.

2. Exam Management

Issuing exams is a capability that the Immediate Assist application currently does not provide. Additional functionality added to the course catalog module which allows the operator to issue exams.

3. Diploma and Certificate Issue

Diploma and course completion certificates are generated when the grading program is executed upon a student obtaining a passing exam score. When a student fails to receive a completion certificate or diploma, the immediate assist clerk should have the ability to immediately initiate re-issue action. This capability should be built into the next iteration of the prototype.

4. Transcript Generation

Currently, transcripts must be requested in writing and are manually prepared. An application to generate a formatted report reflecting transcript information could easily be developed and included in the next iteration of the prototype.

5. Program Enrollment

The prototype now provides the ability to enroll students into individual MOS courses. The ability to enroll or disenroll a student into/from a Professional Military Education (PME) program should be incorporated into a future iteration.

6. LOGAIS Communication

Currently, on-hand quantity of each course component must be manually entered into MCIAIS by the Logistics Department. Automatic stock level data must be incorporated into a future iteration of the application. This does not have to be automated initially, but would allow the Logistics Department to enter into the MCIAIS system a warning that stock levels of a particular course or program are low. Allowing the telephone clerks the ability to see an online warning would eliminate the need for a second call from the student.

7. Help Desk

The ability to record communications between customers and clerks would benefit MCI. Automatic entries into a trouble call table to capture the nature of the call, the action taken, and reference dates would provide an audit trail valuable in the tracking of discrepancies. These items could be incorporated into a help desk module and should be incorporated into a future application upgrade.

Help Desk design should also incorporate interdepartmental communications or E-mail capability. Currently, clerks rely on manually generated "Chits" where clerks annotate the chits with the actions that need to be completed by the MIS, Operations, and Logistics departments. These chits are routed manually and destroyed after the action is completed. There is no audit trail of the request or the actions taken. These efforts could be formalized and incorporated into the help desk module to decrease handling time and increase SSD productivity.

C. ACHIEVEMENT OF RESEARCH OBJECTIVES AND QUESTIONS

This research is the result of a year long project commissioned by MCI to develop the architecture and supporting migration plan to transition from a closed, non-relational system to an open, client/server based relational database management system (DBMS). The research and development was conducted by a five member team at the Naval Postgraduate school. The objectives of this research were achieved: GUI standards were identified and tailored modified specifically for MCI's use, a prototype demonstrating the capabilities of a RDBMS was generated and usability testing to demonstrate the functionality of the prototype and RDBMS was completed.

During this course of this project, the research questions posed in Chapter I were answered. Those questions and their answers are outlined below:

- **Can a clear set of GUI standards for the development of a series of integrated applications be identified?**

GUI standards can be clearly defined and utilized by an application development team at MCI. Weinschenk and Yeo guidelines provide a solid baseline for the application design effort. Application development efforts based on these guidelines will result in applications that require less training for their implementation.

- **Can a proof-of-concept prototype based on the identified GUI standards be developed based on the process and data models created by the NPS project team?**

The "Immediate Assist" prototype is based on the GUI standards delineated by (Weinschenk and Yeo 1995). The prototype demonstrates the capabilities of the models by:

- Entering students into the database,
- Enrolling students into courses,
- Displaying an online course catalog,
- Providing for the offline ordering of materials,
- Retrieving and displaying student history data.

These tasks are presented in accordance with the precise process specifications as outlined in the data model.

Subsidiary questions answered are:

- **Can an object-oriented visual-based application development tool be used to generate a prototype using the GUI standards developed for MCI?**

The "Immediate Assist" prototype was created with Oracle's Developer/2000 IAG. The visual-based OOP tool set, built into the IAG, provided an ideal environment for the development and implementation of this prototype. The MCI project manager received a demonstration of the tool, as well as the prototype, and recognizes the advantages of developing future applications with Developer/2000.

- **Can Usability testing determine whether the prototype meets the needs of the MCI telephone support clerks?**

An exploratory usability test conducted at MCI in a simple single room setup validated the concepts the prototype was designed to prove. With only a three minute

familiarization demonstration, all thirteen participants successfully completed the tasks scripted for the test. Debrief questionnaires conveyed that all participants thought the application was very easy to use and met or exceeded their expectations.

- **Can the process and data models be successfully transformed into a working prototype demonstrating the functionality of the proposed RDBMS?**

The prototype's functionality was predicated on process specifications and data flow diagrams produced by the process modeling efforts. The Immediate Assist prototype successfully manipulates the data contained in the RDBMS.

D. CONCLUSIONS

This research demonstrated:

- GUI Standards can be clearly defined
- Applications based on those standards are easier to use, require less training for the end user, and facilitate the integration of other applications based on the same standards.
- Testing can provide management with feedback that enhances application design and implementation decisions.

Weinschenk and Yeo, provide a set of guidelines that should be used for future application development efforts. These standards provide the application developer a vehicle to make each application look and feel similar by using components in a consistent manner. This consistency allows users familiar with components presented in a previous application to intuitively understand a newly introduced applications.

Application development efforts should include a feedback mechanism. Usability testing incorporated in the development cycle can provide effective feedback. Feedback is necessary throughout the development process beginning with the requirements identification and continuing through application implementation. End-user involvement in the design process ensures that the final product can be understood and used by the operator. "User-centered" testing integrates this feedback into the design process.

Application development firms often have elaborate testing facilities and specialized personnel dedicated to usability testing. When resources preclude the ability to conduct testing under these conditions, smaller tests may be conducted by the organization that are just as beneficial. This thesis provides the necessary information for any organization to establish and conduct its own usability testing.

APPENDIX A. SCREEN SHOTS

This appendix contains selected screen shots of MCI's current DOS-based application along with the prototype GUI-based screen shots for the reader to compare and contrast.

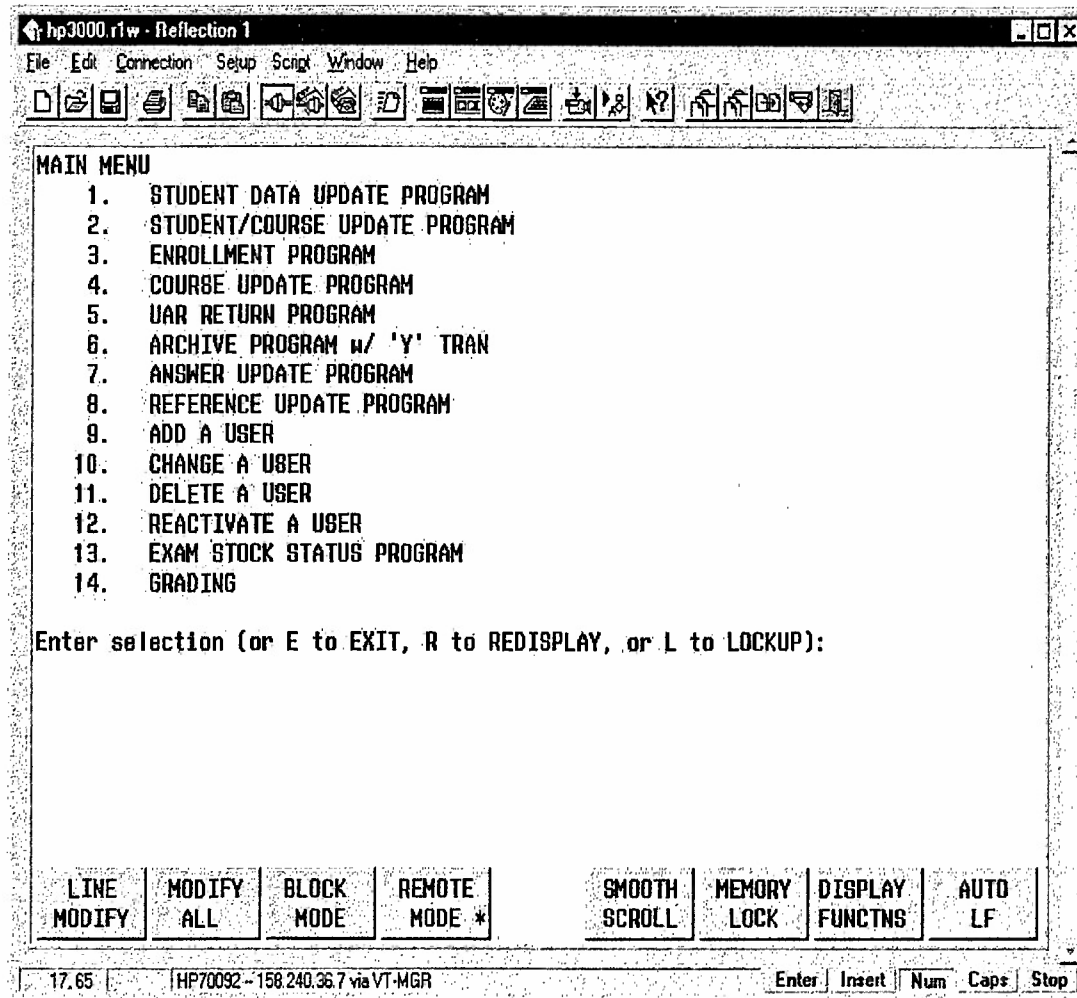


Figure A.1 DOS application Main Screen.

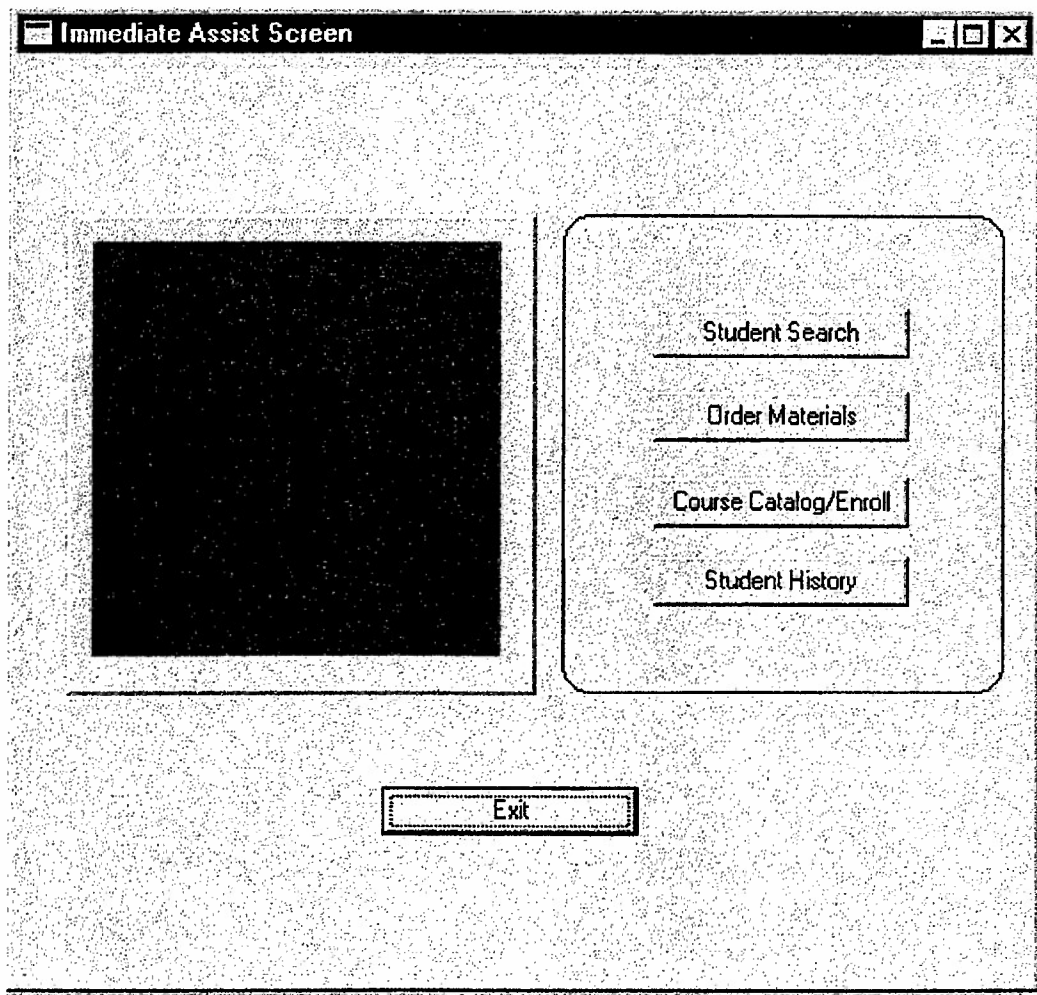


Figure A.2 GUI prototype Main Screen.

hp3000.r1w - Reflection 1

File Edit Connection Setup Script Window Help

On-Line Enrollment Process
Marine Corps Institute
User: ROBNETT

Version: 1.0 Beta Date: 970506

SSN []	1 - USMC Active Duty
Course []	2 - USMC Reserve
Component []	3 - MSG Bn Marine
RUC [] MCC []	4 - USMC Retired
Transaction []	A - Army Active Duty
	5 - Army Reserve
	F - USAF Active Duty
	6 - USAF Reserve
	N - Navy Active Duty
	7 - Navy Reserve
	C - Coast Guard Active Duty
	8 - Coast Guard Reserve
	G - National Guard
	K - Foreign Armed Service
	M - Midshipmen / Cadet
	V - Civilian

Last Name []
Grade []
Rank [] MOS []
Address []
City []
State []
Zip Code []
Update Address? []

Help Backup Clear Enroll Exit

7:15 HP70092 - 158 240.36.7 via VT-MGR Enter Insert Num Caps Stop

Figure A.3 DOS application Enroll Screen.

Immediate Assist Screen

LCDR Hehe

Course 30-MAY-97

Course Number: 001

Title: THE PRINCIPLES OF INSTRUCTION FOR THE MARINE NCO

Date Opened: 12-MAY-89

Date Closed:

Description: Provides NCOs with a basic background of learning theory, techniques, and procedures that prepares them to become effective instructors.

Designed For: LCpl through Sgt in all MOSs

Study Hours: 12

Details....

Enroll in Course

Return to Main screen

Figure A.4 GUI Prototype Enroll/Course Catalog Screen.

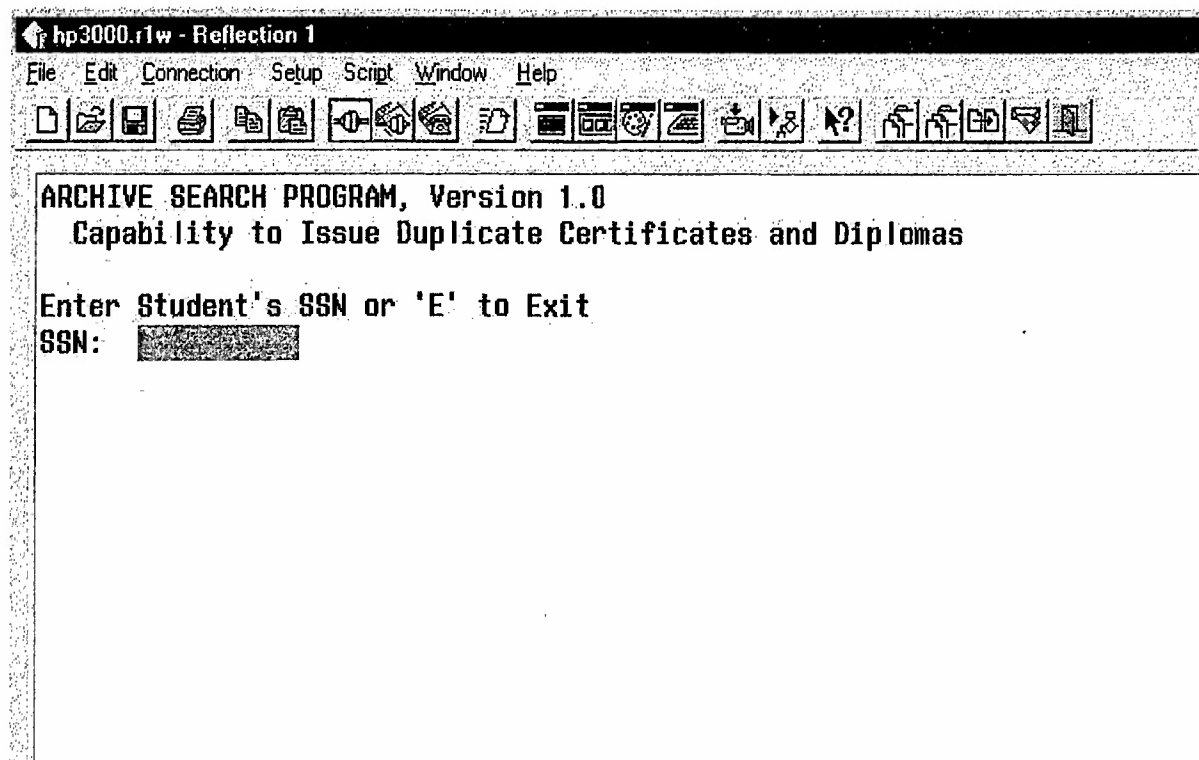


Figure A.5 DOS application Archive Screen.

Immediate Assist Screen

PII Redacted

SSN:

Last Name:

Hehe

First Name:


Gerald

Middle Initial:

L

Rank:

LCDR



Course Data

Course Number	Enrollment Date	Completion Date	Grade	Certificate Date	Disenrolled
8003	15-MAY-97	15-MAY-01	92	27-MAY-97	N
8004	15-MAY-97	15-MAY-01			N
8005	15-MAY-97	15-MAY-01			N
8001	20-MAY-97	20-MAY-01			N
8002	20-MAY-97	20-MAY-01			N

Return

Figure A.6 GUI Prototype Student History Screen.

hp3000.r1w - Reflection 1

File Edit Connection Setup Script Window Help

Marine Corps Institute
Student Address Update Program

Date: 970506 User: ROBNETT

ACTION:

SSN []

Lastname [] Rank [] Grade []

Component [] MOS []

RUC [] MCC []

Address []

City []

State [] Zip Code []

Update Address? []

Backup Clear Form Exit

6.12 HP70092 -- 158.240.36.7 via VT-MGR Enter Insert Num Caps Stop

Figure A.7 DOS Application Student Update screen.

Immediate Assist Screen

Update Student Data

PII Redacted

SSN: [REDACTED]

Last Name: Hehe

First Name: Gerald

Middle Initial: L

Rank: Lieutenant Commander

Grade: 04

Service Component: Navy

Update Data Cancel Update

Figure A.8 GUI prototype Student Update screen.

Immediate Assist Screen

PII Redacted

Student Record

SSN:

Last Name:


SLAUGHTER

First Name:

AARON

Rank:

MAJ



USMC active duty

Address

Address:

221 METZ ROAD

City:

Seaside

State:

CA

Zip Code:

93955

Enrollment

New Student Search

Admin Action

Student History

Order Materials

Main Screen

Figure A.9 GUI Homebase Screen.

Immediate Assist Screen

Customer Data

SSN: [Redacted]

Last Name: [Slaughter]

First Name: [Aaron]

Middle Initial: [T]

Phone

Commercial: [4086562052]

DSN: [8782052]

Address:

Address: [221 Metz Road]

City: [Seaside]

State: [California]

Zip Code: [93955]

Enter/Update Customer

Course Material

NCO Training Materials

Job Aids

Return to Main screen

PII Redacted

Figure A.10 GUI Order Material Customer screen.

Immediate Assist Screen

13-MAY-97

Invoice Number: 5

Customer Number: 486840035

Mailing Address:

Aaron Slaughter

221 Metz Road

Seaside CA

93955

Item Name:	Item #:	Quantity On-hand:	Order Quantity:
R-1 Card	5	2000	34
MACHINEGUN SECTION CHART	BDG06	2345	66

Issue Invoice

Cancel Order

Figure A.11 GUI Prototype Invoice screen.

Immediate Assist Screen

Enrollment Confirmation

Course: 001A

Student SSN: [Redacted]

Date of Enrollment: 05-JUN-97

Required Completion Date: 05-JUN-01

Enroll Cancel

PII Redacted

Figure A.12 GUI Prototype Enrollment Confirmation screen.

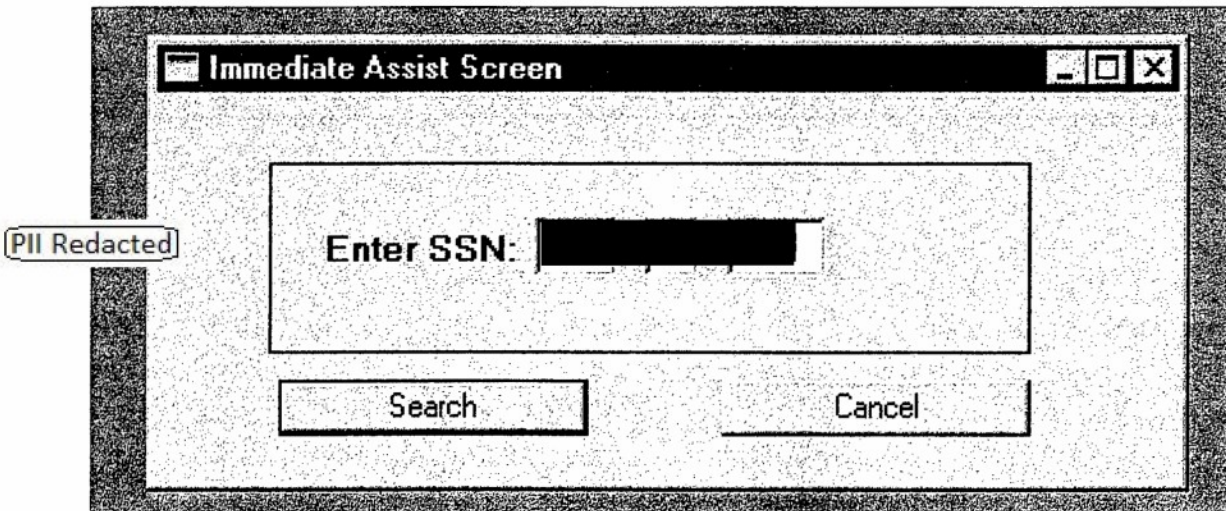


Figure A.13 GUI Prototype SSN Capture screen.

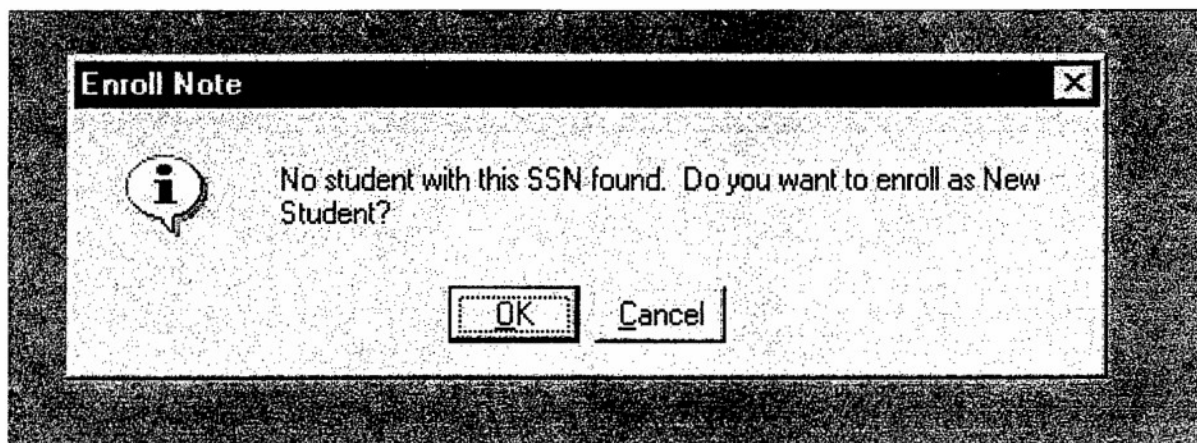


Figure A.14 GUI Prototype Enrollment Dialog Box screen.

APPENDIX B. MCI TEST PLAN

The following test plan was submitted to MCI for approval. Testing was conducted June 10-13, 1997 at MCI. Results are provided to MCI project Manager, Dave Robnette upon compilation including is an executive summary outlining the results.

A. TEST PLAN FOR IMMEDIATE ASSIST PROTOTYPE

1. Purpose

The purpose of this test is two fold. 1) To determine if the application design, screen layout, and the task flows are intuitive and easily understood by the participants, and 2) does the application allow the participants to successfully accomplish the scripted functions.

2. Problem Statement

Can participants successfully find a student present in the database, enter a new student, enroll a student in an existing course, display details about the course, update student data in the database, and retrieve student historical data?

B. TEST TYPE

The test is an assessment test. Its objective is to evaluate the conceptual capabilities of a possible system or application. The results of this test are forwarded to MCI programming personnel for their use in future application design. Specifics include:

- Greet participant -- reinforce the fact that the participant is not being evaluated but is assisting in the evaluation of the application.
- Provide a familiarization of the application to the participant -- enroll, administration update, course data display, history display.
- Administer scenario -- encourage participants to speak out loud and verbalize their understanding of the task and steps towards solving it.

- Administer survey/questionnaire -- encourage opinions and recommendations from participant.
- Debrief participants -- relate what areas they noticed might need improvement or further clarification based on the participant's ability to utilize the prototype.

C. TEST ENVIRONMENT

Based on the available equipment and facilities, the best environment for conducting the usability test is the simple single room setting (Rubin, 1994). This allows the participant and test monitor to interact and provides limited observation opportunities for interested parties. Video equipment, computer, and test materials are provided by testing personnel.

Items requested from MCI are: standard computer keyboard, computer monitor (consistent with the monitors used by the participants during daily operations), and a testing room with limited traffic that can be controlled by test personnel.

D. SELECTING TEST PARTICIPANTS

All available SSD telephone assistance clerks participate in the test. Daily operations were not disrupted. Participants were needed for less than twenty minutes during the conduct of the test.

E. PREPARING TEST MATERIALS

The test staff positions materials prior test commencement and resets materials between participants. At a minimum the staff:

- Ensures the database has been purged of the data entered by the previous participant.
- Ensures the room is completely set up and the prototype launched and ready prior to the participants' arrival.
- Uses the script for the familiarization and the actual test scenario.
- Ensures survey and pencils are available.

F. CONDUCT OF THE TEST

The actual test is administered by having the participant sit directly in front of the keyboard and computer screen. The test monitor sits slightly behind and to the participant's left. A video recorder is set up on the participant's right and focused on the screen. Both the test monitor and the video camera need to be outside of the participant's peripheral vision. Every effort to reduce the anxiety generated by the video device and test monitor's presence, is taken. Prior to the actual conduct of the test, team members verify the setting and make the participant as comfortable as possible. The following Scenario script is used:

- [PII Redacted] 1. LCDR Hehe SSN (), calls....requesting to know if the results of his exam for Course 8003 have been entered and if his diploma has been mailed?
2. LCDR Hehe informs you he has been promoted to Commander.
- [PII Redacted] 3. Lcpl Joe Sample (), training NCO called to order some training aids. His DSN # is 878-2234. Commercial phone is (408)555-1233. His mailing address is Building 6 Room 345, Monterey, CA, 93940. He wants to order 100 R1 cards and 200 R5 cards.
4. Enroll the following students in course 0128, General Personnel Administrations for Reserves. Ask how many reserve credit hours are awarded for this course.
 - [PII Redacted] a. Jeffery A. Gilmore (), Airman,E3,USAF,124 Rickets Rd apt13 Sacramento, Ca,88765
 - [PII Redacted] b. Susan B. Anthony (), Spec4,E4, Army,3345 Iwo Jima Drive, Yuma, Az .85364
 - [PII Redacted] c. Robert D. Jones (), Staff Sergeant, E5, Army, 223 Powell Lane, Tucson Az 85888
5. Staff Sergeant Jones would like to know if he will be awarded any reserve duty credits for taking the general personnel administration for reserves Course 0128.

G. DEBRIEF

The debrief consists of having the participants complete a questionnaire. The questionnaire requests the participants to verbally analyze their thoughts regarding the prototype, specifically commenting on screen layouts, component labels, and the logical flow of the tasks.

Extracting comments from the participant is extremely important. Encouraging the participant to annotate any recommendations on the questionnaire in the "other comments" section is vital.

H. COMPILING DATA INTO FINDINGS AND RECOMMENDATIONS

Data collected from the questionnaire is then summarized and quantified. Questions that request comments are grouped and included with the final report. Video playback will also provide feedback. This data will be transcribed and include in the report.

The results of the study are compiled into an executive study and are made available to the MCI MCAIS project manger for his use during future application design efforts.

Immediate Assist Evaluation Questionnaire

Please circle the number that most appropriately reflects your feeling about the statements. Ask the monitor to show you the area of the prototype the question addresses if you think they will help you better answer the question.

1. Finding a student in the database was an easy task.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

2. Entering a new student into the database was an easy task.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

3. Obtaining information about a class was easy.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

4. Viewing Student history data was an easy task.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

What did you like best about the prototype?

What did you like the least about the prototype?

5. The buttons provided the functionality I expected when I clicked them.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

6. The labels associated with the fields clearly identified the data presented.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

7. A series of applications like this one that covered all the areas I perform daily would be useful.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

8. Comparing the time it took me to learn to use this application to the time it took me to learn my current system, This application was much easier to learn.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

What additional areas or functions would you recommend be incorporated into this application?

If I were designing this application, I would add...

9. This application and others like it would make my job more interesting and improve my ability to relay information to customers.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

10. Do you think that the prototype displays information in a clear enough manner that customers might be able to be able to use it via the internet if access was available?

11. I liked this application.

Strongly agree	Somewhat agree	Not sure	Somewhat disagree	Strongly disagree
1	2	3	4	5

APPENDIX C. USABILITY TEST RESULTS FOR THE MARINE CORPS INSTITUTE

This appendix is a compilation of the data along with a short executive summary of the results of the usability testing conducted at MCI on June 10-11, 1997

A. EXECUTIVE SUMMARY

The results of the usability testing conducted at MCI on June 10-11, 1997, clearly indicate that the Graphical User Interface application design can improve personnel performance while minimizing training costs.

After a three to five minute familiarization session, all thirteen of the test participants were able to accomplish the task scenario without assistance. Effort expended to develop a series of integrated applications based on the process and data models created by the Naval Postgraduate School Marine Corps Institute project team are very beneficial to MCI.

B. TEST DATA

The type of test conducted was an assessment test utilizing a simple single room setup. The purpose of the test was to determine if the application design, the screen layout, and the task flows are intuitive and easily understood by the participants. Specifically, the test determined if the participant could;

- Find a student present in the database
- Enter a new student
- Enroll a student in an existing course
- Display details about the course
- Update student data in the database
- Retrieve student historical data.

C. STATISTICAL DATA

Thirteen of seventeen Marines normally assigned duties as customer support telephone personnel participated in the testing. The questionnaires they completed as well as verbal comments made during the test (recorded on video) and during the debriefing have been compiled in the following paragraphs.

The questionnaire provided a combination of open-ended questions and closed questions. The open-ended questions allowed the participant to elaborate on deficiencies, recommend enhancements, or communicate approval of the implemented functionality. Each question will be provided along with a list of comments or the average grade as calculated from the Likert scale that was attached to each closed question.

1. Finding a student in the database was an easy task.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree

.....1V.....2.....3.....4.....5.....

Average answer value was 1.0. All 13 participants responded "1-Strongly agree" that finding a student in the database was very easy in the prototype.

2. Entering a new student into the database was an easy task.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree

.....1...V.....2.....3.....4.....5.....

Average answer value was 1.15. Participants strongly agree entering a student into the database was very easy.

3. Obtaining information about a class was easy.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree

.....1.....V.....2.....3.....4.....5.....

Average answer value was 1.30. Participants strongly agree obtaining information about a class was easy.

4. Viewing Student history data was an easy task.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree

....1...V.....2.....3.....4.....5.....

Average answer value was 1.15. Participants strongly agree viewing student history is an easy task.

5. What did you like best about the prototype?

Comments consisted of:

- Very reliable
- Volume of information reduces use of references
- Saves time
- Finds records quickly
- Don't lose key data when changing screens
- Makes our job easier
- Easy access
- Everything explained clearly
- Questions answered without making us use our logbooks
- This will definitely put MCI into the future
- Easy program to work with and is convenient
- I liked the program very much
- New info on finding a course is extremely useful
- Using this program you don't have to leave your computer to get information
- Speed and knowledge built in will save us a lot of paper work
- I like the convenience of being able to click on a field and enter the data

6. What did you like the least about the prototype?

Comments consisted of:

- It would be easier to have just an enrollment icon rather than check a social security number
- Some of the categories are vague
- In the course data, list the transactions we have performed
- This was a controlled test drive, so there weren't any kinks
- I have one complaint, that it isn't mine right now.

7. The buttons provided the functionality I expected when I clicked them.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree
....1...V.....2.....3.....4.....5.....
Average answer value 1.07. Participants strongly agree.

8. The labels associated with the fields clearly identified the data presented.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree
....1.....V.....2.....3.....4.....5.....
Average answer value 1.61. Most participants found the labels adequate. However, a large number (6 of the 13) marked this category in the somewhat agree or below area. This area is the most criticized area in the verbal debrief as well.

9. A series of applications like this one that covered all the areas I perform daily would be useful.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree
....1...V.....2.....3.....4.....5.....
Average answer value 1.15. Participants strongly agreed on this area. Verbal debriefs also provided very strong support for developing applications based on this design.

10. Comparing the time it took me to learn to use this application to the time it took me to learn my current system, this application was much easier to learn.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree
....1.....V.....2.....3.....4.....5.....
Average answer value 1.61. Three participants marked the prototype down in this area. Two of the three indicated that this prototype provided too many decisions to make and that the information provided was more comprehensive and therefore a little more complicated to initially understand. The third thought the point and click design did not lend itself to the transaction code he had become used to. Ten participants selected

strongly agree and during the debrief remarked that the prototype was exactly the type of application they would like to have in place.

11. What additional areas or functions would you recommend be incorporated into this application?

Comments included:

- No opinion
- Using course information or title to find a course number
- Add issue exam capability
- Add an area that allows operators to enter data on previous phone calls that can be recalled for historical purposes.
- Issue certificates from the course data screen
- Bigger screen for current course data

11. If I were designing this application, I would add...

Comments included:

- Ability to change address for active duty Marines
- Ability to produce transcripts
- A few games
- Nothing, this application is well thought out and planned.

9. This application and others like it would make my job more interesting and improve my ability to relay information to customers.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree
....1V.....2.....3.....4.....5.....

Average answer value 1.07. Participants strongly agree that a set of integrated applications from this design concept would be more interesting and improve their performance. Verbal debriefs reflected the unanimous support for further application design which would integrate with applications similar to the prototype.

10. Do you think that the prototype displays information in a clear enough manner that customers might be able to be able to use it via the internet if access was available?

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree
.....1.....V.....2.....3.....4.....5.....

Average answer value was 1.53. One participant felt strongly that customers should not have access to the material via the internet marked strongly disagree. During the debrief he indicated that the application design probably could be used on the internet but as a policy it should not be allowed. This mark could be removed as an answer out of context from the question. If removed the modified value becomes 1.23 and indicates that the participants agree that the prototype displays the information clearly enough to allow customers to access the data via the internet.

11. I liked this application.

Strongly agree Somewhat agree Not sure Somewhat disagree Strongly disagree
.....1V.....2.....3.....4.....5.....

Average answer value 1.0. All participants like this application and felt it could help them perform their duties faster, more accurately and also provided a more interesting interface in which to manipulate data

APPENDIX D. DEFINITIONS AND ABBREVIATIONS

AIS	Automated Informations System
BAA	Business Area Analysis
BPR	Business Process Reengineering
BSD	Business System Design
CASE	Computer Aided Systems Engineering
CD	Compact Disk
CDE	Cooperative Development Environment
CPU	Central Processing Unit
CRT	Cathode Ray Tube
DOS	Disk Operating System
DFD	Data Flow Diagrams
EAP	Enterprise Architecture Planning
GUI	Graphical User Interface
HP	Hewlett Packard
ITM	Information Technology Management
IAG	Interactive Application Generator
ILT	Instructor Led Training
ISP	Information Strategy Planning
LOV	List of Values
MCI	Marine Corps Institute
MCLAIS	Marine Corps Institute Automated Information System
MCTFS	Marine Corps Total Force System
MOS	Military Occupational Specialty
MS	Microsoft
NPS	Naval Postgraduate School
OOP	Object Oriented Programming
PC	Personal Computer
PME	Professional Military Educational
PL/SQL	Program language/Structured Query Language
RAD	Rapid Application Development
RDBMS	Relational Database Management System

SSD	Student Services Department
SSN	Social Security Number
SQL	Structured Query Language
TD	Technical Design
TFS	Total Force System

LIST OF REFERENCES

Baden, Kurt A., and Peters, Gerald A., A Business Process Model and Reengineering Plan for the Student Services Department of the Marine Corps Institute, Master's thesis, Naval Postgraduate School, Monterey, California :September, 1997

Galitz, Wilbert O., It's Time to Clean your Windows, John Wiley and Sons, Inc. New York, New York :1994

Galitz, Wilbert O., User-Interface Screen Design, John Wiley and Sons, Inc. New York, New York :1993

Kendall, Kenneth E. and Kendall, Julie E., Systems Analysis and Design ,Prentice Hall, Englewood Cliffs, New Jersey :1995

Kroenke, David M, Database Processing, Fundamentals, Design, and Implementation, Prentice Hall, Englewood Cliffs, New Jersey :1995

Lulushi, Albert, Developing Oracle Forms Applications ,Prentice Hall, Englewood Cliffs, New Jersey :1996

Martin, James, Information Engineering, Book II, Planning and analysis ,Prentice Hall, Englewood Cliffs, New Jersey :1990

Mueller, Robert J., Oracle Developer/2000 Handbook, Oracle Press/McGraw Hill, Berkeley, California :1996

Slaughter, Aaron, A Relational Database Model and Data Migration Plan for the Student Services at the Marine Corps Institute, Master's thesis, Naval Postgraduate School, Monterey, California :September, 1997

Spewak, Steven H., Enterprise Architecture Planning (EAP), Developing a Blueprint for Data, Applications and Technology John Wiley and Sons, Inc. New York, New York :1992

Weinschenk, Susan and Yeo, Sarah C., Enterprise-Wide GUI Design, John Wiley and Sons, Inc. New York, New York :1995

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
 John J. Kingman Rd., STE 0944
 Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library2
 Naval Postgraduate School
 411 Dyer Rd.
 Monterey, CA 93943-5000

3. Director, Training and Education1
 MCCDC, Code C46
 1019 Elliot Rd.
 Quantico, Virginia 22134-5027

4. Director, Marine Corps Research Center.....2
 MCCDC, Code C40RC
 2040 Broadway Street
 Quantico, Virginia 22134-5107

5. Director, Studies and Analysis Division.....1
 MCCDC, Code C45
 300 Russell Road
 Quantico, Virginia 22134-5130

6. Marine Corps Representative1
 Naval Postgraduate School
 Code 037, Bld. 234, HA-220
 699 Dyer Road
 Monterey, CA 93943-5000

7. CDR Gerald L. Hehe2
 4609 Brideshead Ct.
 Virginia Beach, VA 23464

8. Professor Magdi H. Kamel, Code SM/Ka.....2
 Department of Systems Management
 Naval Postgraduate School
 Monterey, CA 93943-5000

10. The Marine Corps Institute2
Attn: Maj. Lloyd J. Hamashin, USMC
Washington Navy Yard, 912 Poor St. S.E.,
Washington, DC 20391-5680
11. LCDR Dale Courtney, USN, Code 05C1
Department of Computer and Information Services
Naval Postgraduate School
Monterey, CA 93943-5000